

Saxphone Player Edition

SPAWN

P3rL CockBook SaXaphOnE PlaYer ThEory

Perl Strings OverView

لغة البيرل مثل جميع اللغات البرمجية الاخرى التي تحتوي على السلاسل النصية او ما يعرف في الوسط البرمجي بال strings
غالبا ما اخذنا على هذا المصطلح على انه السلاسل النصية يجب ان تكون محصورة بين علامات الاقتباس هذه المعلومة صحيحا ولكن في لغة البيرل من الممكن ان يتم طباعة سلسلة نصية من غير وجود علامات الاقتباس اليكم الطرق التي من الممكن ان يتم طباعة السلاسل النصية فيها وهي كما يلي في هذا الكود

Code(1)

```
$a="www.programming-fr34ks";
$b='programming-freaks';
$c=securitygurus;
print $a,"\n",$b,"\n",$c,"\n";
```

وعند تنفيذ الكود هذا ستكون هذه الصورة هي ناتج التنفيذ كما يلي

```
$a="www.programming-fr34ks";
$b='programming-freaks';
$c=securitygurus;
print $a,"\n",$b,"\n",$c,"\n";
www.programming-fr34ks
programming-freaks
securitygurus
```

Figure(1)

هذا هو ناتج تنفيذ الكود الموجود في الاعلى عن طرق طباعة السلاسل النصية
وفي الصفحات القادمة سوف نتعمق في برمجة السلاسل النصية واساليب عملها المختلفة

Perl Substrings

السلسلة النصية هي عبارة عن كلمة او جملة او مجموعة من الكلمات التي تشكل السلسلة النصية

المتكاملة ولكن ماذا لو أراد احد أن يعمل access على جزء من السلسة النصية ففي هذه الحالة لابد من ايجاد طريقة تساعدنا على عمل ما نحتاج من تحرير او طباعة او استبدال في السلسلة النصية لذا

في وضع كهذا الوضع الحل الانسب هو يتم بأستعمال دالة ال substr ويكون تمثيل هذه الدالة برمجيا هو كما يلي

code(2)

```
$a="Hi,Iam perl programmer";
$b=(substr($a,4));
print $b;
```

هذه هي احدى طرق استعمال هذه الدالة نلاحظ انه عند تنفيذ هذا الكود يكون ناتج التنفيذ الناتج عنه هو كمايلي في هذه الصورة

```
$a="Hi,Iam perl programmer";
$b=(substr($a,4));
print $b;
am perl programmer
```

Figure(2)

لو تدقق في الصورة التي في الاعلى تلاحظ انه عند بدء الطباعة ان البيرل قامت بطباعة السلسة النصية الموجودة في البرنامج بداية من الحرف الرابع لانه تم اخباره ان يقوم بذلك ولو تم اعلام البيرل بغير رقم لقامت بعرض نتيجة اخرى غير هذه النتيجة هذا الوضع الاول من اوضاع تمثيل هذه الدالة الان لاحظ الوضع او التمثيل الثاني الخاص بهذه الدالة وهو كما يلي في الكود التالي

*code(3)

```
$a="Hi,Iam perl programmer";
$b=(substr($a,4,10));
print $b;
```

```
$a="Hi,Iam perl programmer";
$b=(substr($a,4,10));
print $b;
am perl pr
```

Figure(3)

الان عند تنفيذ هذا السكريبت البرمجي تلاحظ انه عند التنفيذ سيعرض الناتج الاتي
لاحظ انه هنا اهم فقرة يجب عليك ان تعرفها هي انه الرقم عشرة هنا يعامل على انه offset
ولو دققت الملاحظة في البرنامج تلاحظ انه تم عد عشر حروف من موقع الحرف الرابع الى ما بعده بعشرة حروف
هذا ايضا كان التمثيل البرمجي الثاني وهنالك ايضا تمثيل برمجي أخر لهذه الدالة وهو كما يلي

Code(4)

```
$a="Hi,Iam perl programmer";
$b=(substr($a,7,4)) = "Fr34k";
print $a;
print "\n";
```

هذا كان التمثيل البرمجي للوضع الثالث الخاص بهذه الدالة ونلاحظ ان الذي حدث في هذا السكريبت البرمجي هو انه تم استبدال الكلمة بيرل التي تكون مبدوئة بالحرف سبعة في الجملة ويكون طولها هو بالضبط 4 احرف تم استبدالها بالكلمة التي تم تحديديها في الفقرة الاخيرة من السطر البرمجي الثاني الذي وهنا يتم اعتبار كلمة Fr34kعلى اساس انها newstring
ولو نفذت البرنامج ستكون ناتج العملية هو الأتي في الصورة

```
$a="Hi,iam perl programmer";
$b=(substr($a,7,4))="Fr34k";
print $a;
print "\n";
Hi,iam Fr34k programmer
```

Figure(4)

أما عن التمثيل الرابع فهو يختلف تماما عن الاوضاع التي تم ذكرها قبل قليل وهي كما يلي

Code(5)

```
$a="HI,iam perl programmer";
$b=(substr($a,-10));
print $b,"\n";
```

قبل الكلام عن ناتج تنفيذ هذا البرنامج فأن هذه الطريقة غالبا ما يتم أطلاق عليها اسم طريقة القراءة العكسية وناتج تنفيذ هذه الطريقة هو كما ظاهر في هذه الصورة

```
$a="HI,iam perl programmer";
$b=(substr($a,-10));
print $b,"\n";
programmer
```

Figure(5)

How to not use Temp Values

لو كنت تريد ان تحول القيمة الموجودة في المتعير الاول الى متغير أخر ولكن من دون استعمال متغير ثالث على أساس اعتباره مخزن مؤقت لحل هذه المشكله هذه فأن اغلب المبرمجين يتبعون هذه الطريقة الموجودة في هذا الكود ألاتي

Code(6)

```
$a="Hi,iam perl programmer";
$temp=$a;
$b=$a;
$b=$temp;
print $b,"\n";
```

ولكن كما ذكرنا فأن هذه الطريقة هي طريقة غير مرغوبة وهي ايضا طريقة مزعجة لذا فأنه من الممكن استعمال طريقة أخرى تكون الطريقة الاخرى خالية من أي استعمال للمتغيرات المؤقتة ويمكن ان يتم عمل ما ذكر اعلاه في هذا الكود البسيط

Code(7)

```
$a="programming-fr34ks";
$b="securitygurus";
($a,$b)=($b,$a);
print $a,"\n",$b,"\n";
```

لاحظ انه الكود هو كود سهل لايحتاج الى توضيح وناتج تنفيذه هو كما الاتي في الصورة المدرجة ادناه

```
$a="programming-fr34ks";
$b="securitygurus";
($a,$b)=($b,$a);
print $a,"\n",$b,"\n";
securitygurus
programming-fr34ks
```

Figure(6)

Converting Between Characters And Values

هذه العملية تتم من خلال استعمال نوعين من الدوال هذه الدوال هي:-

1- chr
2- ord

حيث تعمل الدالة الاولى على تحويل الرقم الى قيمة حرفية
بينما تعمل الثانية على تحويل الحرف الى قمية عددية ويكون التمثيل البرمجي الخاص بهاتين الدالتين هو كما يلي

Code(8)

```
$a=115;
$b="s";
print (chr($a));
print "\n";
print (ord($b));
print "\n";
```

والان لو نفذت البرنامج التالي ودققت في ناتج التنفيذ سيكون الناتج هو كما يلي في الصورة الاتية

```
$a=115;
$b="s";
print (chr($a));
print "\n";
print (ord($b));
print "\n";
s
115
```

Figure(7)

لاحظ من العملية انه تم عمل convert بين قيم المتغيرات الموجودة في البرنامج
لمجرد التذكير فأن دالة ال ord تستعمل لايجاد قيم ال Ascii للقيم المعطاة لها

Processing a String One Character at a Time

تخيل انه لديك سلسلة نصية مكونة من عدد من الحروف وانت تريد ان تعمل عزل لهذه السلسة النصية بحيث انه في النهاية ستحصل على الحروف الاولية المكونة لهذه السلة النصية من دون تكرار فيها اي بمكن القول انك فلي هذه الحالة ستحصل على المواد الاولية و هي الحروف المكونة للسلسة النصية دون تكرار في هذه الحروف وهذه العملية غالبا ما يتم حلها من خلال التعابير المنتظمة والتي سوف نتطرق اليها فيما بعد ولكن الان سنلجأ الى هذه الطريقة الخالية من فن التعابير المنتظمة و اليكم الطريقة

في البداية سيكون العمل مع دالة ال split مع نموذج pattern فارغ لكي يتم كسر السلسلة النصية الى individual characters
والتمثيل البرمجي لهذه العملية هو كما يلي

Code(9)

```
%rare=();
$string="we are perl propgrammer";
foreach $char (split //,$string){
$rare{$char}++;
}
print "the characters of the $string is ",sort (keys %rare);
```

لو قمت الان بتنفيذ هذا السكريبت البرمجي ستكون النتيجة منه هي كما يلي

```
%rare=();
$string="we are perl propgrammer";
foreach $char (split //,$string){
$rare{$char}++;
}
print "the characters of the $string is ",sort (keys %rare);
the characters of the we are perl propgrammer is  aeglmoprw
```

Figure(8)

الان لو نعمل اوفر فيو على هذا السكريبت تلاحظ فعلا انه تم استعمال الدالة split من أجل ان تكسر السلسة النصية الى individual characters وتلاحظ فعلا انه تم وضع المتغير الذي يحمل السلسة النصية معها في جملة تكرار هذه هي فكرة البرنامج وتجدر الاشارة الى ان البرنامج فكرته مقتبسة من كتاب

*Code(10)

O'Reilly - Perl Cookbook 2nd Edition.chm

Reversing the Strings

كيف من الممكن ان تتم عمليات عكس السلاسل النصية ؟؟
طبعا تتم عمليات عكس السلاسل النصة من خلال استعمال الدالة التي تعمل على عكس المتغيرات وهي دالة ال reverse ويكون تمثيل هذه الدالة برمجيا كما يلي

Code(11)

```
$a="We Are Perl Programmer";
$b=(reverse($a));
print $b,"\n";
```

وعند تنفيذ هذا المقطع البرمجي يكون الناتج من تنفيذه هو كما الاتي

```
$a="We Are Perl Programmer";
$b=(reverse($a));
print $b,"\n";
remmargorP lreP erA eW
```

Figure(9)

وبالاضافة الى هذه الخاصية فأن لهذه الدالة أمكانيات اخرى من الممكن الاستفادة منها ولكن سيتم شرحها في الفصول القادمة ومع الانواع المتغيرات الاخرى

How To Shrink Tabs And How to Expand Spaces

في تعاملنا في كثير من حالات السلاسل النصية قد نصادف ال skip squence the \t
في حالات متعددة قد تسبب هذه الحالات بعض التشويش للمتسخدم لذا من الممكن ان يتم تقليصها من خلال استعمال دالة الاستبدال كما يلي

***Code(12)**

```
$_="Spawn\tis\tperl\tprogrammer";
s/\t/ /g;
print $_;
```

لاحظ في هذا المثال تم استبدال كل رموز الهروب الخاصة بال(\t) الى مجرد فراغات هذه العملية تسمى Compressing tabs
ومن الممكن ان تتم هذه العملية بالعكس اي من خلال الفراغ نعمل عمليات توسع للرمز الهروب ال(\t)
وتسمى هذه العملية expand tabs ويكون تمثيلها البرمجي كما يلي في هذا المثال السكريبت

***Code(13)**

```
$_="Spawn is perl programmer";
s/ /\t/g;
print $_;
```

وهذه الصورة الناتجة من عملية التنفيذ

```
$_="Spawn is perl programmer";
s/ /\t/g;
print $_;

Spawn	is	perl	programmer
```

Figure(10)

احب أن انوه الى فقرة وهي سبب وجود الحرف(g) في نهاية استعمال الدالة الخاصة بعملية الاستبدال هذه الفقرة سيتم ذكرها لاحقا في مواضيع لاحقة

Variables In User Output

هل من الممكن ان يتم استعمال متغير مسند له قيمه سلسلة نصية ان يتم استعمالها مخرجات اليوزر؟؟
في لغة البيرل هذه العملية ممكنة ويكون اسلوب استعمالها البرمجي كما يلي من خلال السكريبت المدرج أدناه

Code(14)

```
$name="Spawn";
$lang="perl";
$age="21";
print "My name is $name,\nmy Favorite G33k lang is $lang\niam $age years old\n";
```

ما هو اساس عمل هذا السكريبت البرمجي ؟؟

في الواقع أن أساس عمله هو سهل ومفيد في السكريبت اعلاه يوجد متغير اسمه $name هذا المتغير يحمل قيمة لسلسلة نصية قيمتها هي spawn وعند تنفيذ البرنامج اخبرنا مترجم البيرل انه اطبع جملة my name is والى جوارها استعمل القيمة المخزونة داخل المتغير الذي يحمل اسم $name هذه هي فكرة السكريبت المذكور في الاعلى وصورة ناتج التنفيذ الخاصة به هي

```
$name="Spawn";
$lang="perl";
$age="21";
print "My name is $name,\nmy Favorite G33k lang is $lang\niam $age years old\n";

My name is Spawn,
my Favorite G33k lang is perl
iam 21 years old
```

Figure(11)

Controlling Case

محور هذا الموضوع هو كيفية التحكم في حالة الاحرف ما بين الكبيرة و الصغيرة و الدوال التي تكون مسؤلة عن هذه العملية

وسوف نتكلم عن جميع الدوال والاساليب التي يتم من خلالها التحكم في الاحرف

1-uc(upper case)

التمثيل البرمجي الخاص بهذه الدالة هو

Code(15)

```
$a="Spawn perl programmer";
print (uc($a));
print "\n";
```

عمل هذه الدالة انها تعمل على تحويل الحروف المكونة للسلسلة النصية من الاحرف الصغيرة الى الاحرف الكبيرة او بتعبير أصح انها تعمل على تحويل جميع الاحرف المكونة للسلسة النصية الى حروف كبيرة

2-ucfirst(upper case first)

يكون التمثيل البرمجي لهذه الدالة هو كما يلي

Code(16)

```
$a="spawn is perl programmer";
print (ucfirst($a));
print $a;
```

عمل هذه الدالة يكون قائم على انه يتم تحويل الحرف الاول من السلسلة النصية الى حرف حالة كبيرة مهما كان اي اذا كان صغير يتحول الى كبير و اذا كان كبير يبقى كما هو

3-lc (lower case)

ويكون التمثيل البرمجي الخاص بهذه الدالة هو كما يلي

Code(17)

```
$a="SpawN is PERL PROGRAmmer";
print (lc($a));
```

هذه الدالة يكون عملها تماما عكس الدالة الاولى أي انها تعمل على عكس حالة الاحرف من الحالة الكبيرة الى الحالة الصغيرة ويكون تطبيقها على جميع الاحرف على عكس الدالة رقم 2 يكون عملها مقصور على الحرف ألاول اي اذا كانت الاحرف كبيرة سوف تتحول الى احرف صغيرة و اذا كانت صغيرة سوف تبقى كما هي

4-lcfirst(lowercase first)

ويكون التمثيل البرمجي الخاص بهذه الدالة كما يلي

Code(18)

```
$a="Spawn is perl programmer";
print (lcfirst($a));
```

اما هذه الدالة فأن عملها يكون محدود على تحويل الحرف الاول من السلسلة النصية الى حرف صغير اذا كان حرف كبير وفقط اي عملها يشبه عمل الدالة رقم 2

Controlling Case (Anchors)

في الصفحة السابقة سبق وان تمت مناقشة اساليب التحكم في حالة الاحرف من خلال الدوال و لكن في هذا لانه سوف نتحكم في حالات الاحرف من خلال ما أحب أن اسميه بأسم ال anchors يشبه عمل ال anchors الحقيقي في التعابير القياسية

1-\U

التمثيل البرمجي الخاص بها هو كما يلي

Code(19)

```
$a="\Uspawn";
print $a;
```

هنا عمل هذا المعرف هو انه يعمل على تحويل جميع الاحرف من الحالة الصغيرة الى الحالة الكبيرة واذا كانت الاحرف حالتها كبيرة يبقيها على ماهي عليه من وضعية اذن عملها من الناحية البرمجية يشبه عمل دالة ال uc

2-\u

ويكون التمثيل البرمجي الخاص بهذا المعرف هو كما يلي

Code(20)

```
$a="\uspawn is perl programmer";
print $a;
```

أما عن عمل هذا المعرف فهو يشبه عمل الدالة ucfirst التي تعمل على تحويل الحرف الاول من السلسلة من السلسة النصية الى حرف كبير

3-\L

عمل هذا المعرف يكون كما يلي

Code(21)

```
$a="\LSpAWN IS PeRL proGraMMer";
print $a;
```

هذا المعرف يعمل على تحويل الاحرف من الحالة الكبيرة الى الحالة الصغيرة اي انه في مجال العمل فأنه يشبه عمل دالة ال lc

4-\l

يكون التمثيل البرمجي لهذا المعرف هو كما يلي

Code(22)

```
$a="\lSpawn is perl programmer";
print $a;
```

اما من ناحية عملها البرمجي فأنها تشبه عمل الدالة lcfirst

How To Use The Index (Regular)

في هذا الموضوع سوف يكون الحيدث عن كيفية ايجاد الموقع الخاص بحرف ما داخل السلسلة النصية وتتم هذه العملية من خلال استعمال الدالة index ويكون تمثيل هذه الدالة برمجيا واسلوب استعمالها هو كما يلي

Code(23)

```
$a="perl";
print (index($a,e));
```

في هذا السطرين البرمجيين تلاحظ انه عن تنفيذ البرنامج سوف تحصل على هذه القيمة وهي

```
$a="perl";
print (index($a,e));
1
```

Figure(12)

والتي تدل على انه الحرف e يحمل الموقع رقم 1 في السلسلة النصية وان ترتيب حروف السلسلة النصية مفهرس صفريا

ملاحظة:-
تجدر الاشارة الى فقرة مهمة للغاية وهي انه حالة الاحرف مهمة هنا تكون مهمة اي انه اذا كانت الحرف الموجود في السلسلة النصية صغير و الحرف المستعمل في الدالة كبيرة فأنه سوف تحصل على خطأ لذا عليك ان تنتبه الى هذه الفقرة

How To Use The Rindex

المحور البرمحي لهذا الموضوع هو كيف يتم ايجاد موقع حرف معين داخل سلسلة نصية ولكن يسأل احد ما ما هو الفرق بين هذا الموضوع و الموضوع الذي يسبقه ؟؟
الاجابة هي انه الفرق يكون في الدالة المسؤولة عن هذا العمل وعن اسلوب القراءة المختلفة الان لاحظ السكريبت البرمجي لكي تعرف ما هو الفرق بين هذين الموضوعين

Code(24)

```
$a="perl programmer";
print (rindex($a,e));
```

اسلوب عمل هذه الدالة البرمجية يشابه نوعا ما عمل الدالة البرمجية الخاصة بالموضوع السابق قبل ان نتكلم عن عمل هذه الدالة اليكم الصورة الناتجة من عملية التنفيذ وهي الصورة ادناه

```
$a="perl programmer";
print (rindex($a,e));
13
```

Figure(13)

هذه الصورة الناتجة عن تنفيذ السكريبت البرمجي أعلاه ولكن ألان سنقوم بتنفيذ هذا البرنامج باستخدام نفس السلسلة النصية ولكن مع استعمال دالة ال index
اليكم الكود

Code(25)

```
$a="perl programmer";
print (index($a,e))
```

الان لو لاحظت ناتج تنفيذ هذا البرنامج لسوف يكون كما يلي في هذه الصورة

```
$a="perl programmer";
print (index($a,e))
1
```

Figure(14)

الناتج من تنفيذ هذا البرنامج هو رقم 1 على عكس ناتج تنفيذ البرنامج في السابق الذي كان ناتج تنفيذه هو رقم 13
على الرغم من أن كل من البرنامجين تم استعمال معهم نفس السلسلة النصية و نفس الحرف ولكن نقطة الاختلاف هي تكمن وعمل هاتين الدالتين هو كما يلي

1-index

تعمل على قراءة السلسلة النصية من البداية الى النهاية حيث انه الجملة النصية المستعملة تحتوي على مرتين حرف
e وفي الدالة اندكس يتم عرضه على انه الرقم واحد لانه ثاني حرف في السلسلة النصية

2-rindex

في الدالة السابقة ذكرنا انه يتم قراءة السلسلة النصية من البداية النهاية هذا لايعني ان الدالة هذه تقرأ من النهاية الى البداية ولكن هذا يعني انها تعمل على عرض اول حرف من الاخير تتم المطابقة عليه ولاحظتم الفرق بين ناتج تنفيذ البرنامجين والفرق بين النواتج

How To Length

كيف تقوم بأيجاد طول سلسلة نصية ؟؟
هذه عملية سهلة وتجد دالة خاصة بهذه العملية حيث ان هذه الدالة تعمل على قياس طول السلسلة النصية وهي دالة ال length ويكون تمثيل هذه الدالة برمجية كما يلي

***Code(26)**

```
$a="perl";
print (length($a));
```

ويكون ناتج تنفيذ هذا الكود هو كما يظهر في الصورة الاتية

```
$a="perl";
print (length($a));
4
```

Figure(15)

How To Reformat

أنت الان لديك سلسلة نصية كبيرة مكونة من عدد من أسطر اي كاثر من سطرين او 3 كيف تقوم بعمل تنسيق نصي لها بحيث تظهر عند التنفيذ على انها نص مهندم من ناحية العرض هذه اعلملية ممكنة في لغة البيرل من خلال استعمال موديل ال Text::Wrap
وتتم العملية كما يلي من خلال هذا الكود

***Code(27)**

```
@Who_We_Are=(      "SpAwN is perl programmer",
                   "StrikerX is Pyhton programmer",
                   "St0rM is C programmer",
                   "Mutati0n is RuBy programmer",
                   "Dj is bash programmer",
                   );
use Text::Wrap qw($columns &wrap);
$columns = 20;
print wrap ("     ","  ",@Who_We_Are),"\n";
```

Over view on the code

الان نأخذ نظرة سريعة على هذا البرنامج لاحظ ما يلي

1-

***Code(28)**

```
@Who_We_Are=(      "SpAwN is perl programmer",
                   "StrikerX is Pyhton programmer",
                   "St0rM is C programmer",
                   "Mutati0n is RuBy programmer",
                   "Dj is bash programmer",
                   );
```

لاحظ هنا في الاعلى لدينا مصوفة مكونة من هذه الاسطر لا أكثر ولا أقل ولن ادخل في برمجة المصوفة لأنه هذا الموضوع سوف يناقش لاحقا

2-

***Code(29)**

```
use Text::Wrap qw($columns &wrap);
$columns = 20;
print wrap ("     ","  ",@Who_We_Are),"\n";
```

اما هنا تم استعمال الموديل الخاص بهذه العملية وقمنا باستعمال متغير يحمل أسم $columns وروتين فرعي يحمل أسم &wrap
ومن اثم المتغير الذي يحمل اسم $COLUMNS

تم أسناد قيمة رقمية له و هي الرقم 20 وهذا الرقم يعني انه كل سطر من هذه الاسطر سوف يكون مكون من 20 حرف على اقصى حد ولن يكون هنالك سطر مكون من اكثر من 20 حرف ولكن من الممكن ان يكون السطر مكون من أقل من 20 حرف هذه الفقرة ممكنة ولكن العكس غير ممكن
أليكم الصورة الناتجة من تنفيذ البرنامج وهي كما يلي

```
    SpAwN is perl
  programmer
  StrikerX is
  Pyhton programmer
  St0rM is C
  programmer
  Mutati0n is RuBy
  programmer Dj is
  bash programmer
```

***F**igure(16)*

ملاحظة هامة
هذا الكود ليس كغيره من الاكواد الاخرى على اعتبار انه يدخل الموديل في جزء برمجي منه
الان اذهب الى هذا المسار

****C**ode(30)*

```
/usr/lib/perl5/5.8.8/Text/
```

وافتحه سترى فيه ملف يحمل هذا الاسم الموجود في هذه الصورة



***F**igure(17)*

اذا كان لديك هذا الملف موجود هذا يعني ان الموديل موجود على جهازك وألا فأنه عليك تحمله من موقع ال www.cpan.org
الان افتح هذا الملف ستلاحظ انه مكون من هذه التركيبة البرمجية تشبه التركيبة الموجودة في الصورة

```
package Text::Wrap;

use warnings::register;
require Exporter;

@ISA = qw(Exporter);
@EXPORT = qw(wrap fill);
@EXPORT_OK = qw($columns $break $huge);

$VERSION = 2006.1117;

use vars qw($VERSION $columns $debug $break $huge $unexpand $tabstop
        $separator $separator2);
use strict;

BEGIN   {
        $columns = 76;  # <= screen width
        $debug = 0;
        $break = '\s';
        $huge = 'wrap'; # alternatively: 'die' or 'overflow'
        $unexpand = 1;
        $tabstop = 8;
        $separator = "\n";
        $separator2 = undef;
}

use Text::Tabs qw(expand unexpand);

sub wrap
{
        my ($ip, $xp, @t) = @_;

        local($Text::Tabs::tabstop) = $tabstop;
        my $r = "";
        my $tail = pop(@t);
        my $t = expand(join("", (map { /\s+\z/ ? ( $_ ) : ($_, ' ') } @t), $tail));
        my $lead = $ip;
        my $ll = $columns - length(expand($ip)) - 1;
        $ll = 0 if $ll < 0;
        my $nll = $columns - length(expand($xp)) - 1;
        my $nl = "";
```


Figure(18)

لو ترجع البرنامج الاول لكنت تلاحظ انه لو غيرت اسم الرويتن الفرعي من الاسم الذي عليه وهو ال wrap
الى اي أسم اخر غير هذا الاسم لتلاحظ انه سوف يكون هنالك خلل في تنفيذ البرنامج ولن تحصل على نتجية
ونفس الوضع مع المتغير الذي يحمل الاسم $columns
لو غيرت اسمه من هذا الاسم الى اسم اخر لن يتنفذ البرنامج ولكن اذا اردت ان تتعلم كيف تتم هذه العملية عليك
ان تغير اسمائهم من الملف الذي اشرنا اليه في الاعلى و ان تكون global
وليس تغير جزئي

How To Use Constant

كيف من الممكن ان تقوم بأسناد سلاسل نصية الى متغيرات وتكون قيم هذه المتغيرات غير قابلة للتبديل لاحظ انه لو كان لديك سكريبت برمجي مثل هذا السكريبت

Code(31)

```
$a="perl";
$a="PERL";
print $a;
```

لو تلاحظ ما هو ناتج تنفيذ هذا البرنامج لسوف يكون كما يلي

```
$a="perl";
$a="PERL";
print $a;
PERL
```

Figure(19)

لاحظ انه البرنامج مكون من متغيرين كلاهما يحمل نفس الاسم المتغير الاول كان يحمل قيمة البيرل في حالة الاحرف الصغيرة ولكن المتغير الاخر يحمل قيمة البيرل لكن في حالة الاحرف الكبيرة وعندما تم تنفيذ البرنامج كان الناتج من هذا التنفيذ هو طباعة كلمة البيرل في حالة الاحرف الكبيرة اذن تم تغيير قيمة المتغير من الاحرف الصغيرة الى الكبيرة اذن هذه العملية غير مفيدة في هذه الوضعية اذن لا بد من أتباع غير اسلوب وهذا الاسلوب يتم من خلال استعمال الموديل constant وكما يلي من خلال هذا الكود

Code(32)

```
use constant WHO_WE_ARE =>"Programming-fr34ks";
print WHO_WE_ARE;
```

الان لو تم ملاحظة ناتج تنفيذ البرنامج فأنه يكون كما يلي في هذه الصورة

```
use constant WHO_WE_ARE =>"Programming-fr34ks";
print WHO_WE_ARE;
Programming-fr34ks
```

Figure(20)

لاحظ هنا في البرنامج الموجود اعلاه ال

WHO_WE_ARE

لايتم معاملتها على أنها متغير من وجهة نظرى ولكن يتم اعتبار ها

لسلسلة النصية nick name

and this nickname is disabled from accessing

ملاحظة

هذا ال module من الممكن ان يتم استعماله مع الارقام اي أنه لايقتصر تعامله مع السلاسل النصية

How To L33t

ألان وفي الوقت الحالي اصبح كل من له اطلاع في عالم البرمجة وحتى لو كان هذا الاطلاع هو اطلاع بسيط على مكونات عالم البرمجة سيعلم ماهو ال leet speak طبعا لغة البيرل هذه الامكانية فيها متوفرة عن طريق استعمال الموديل الخاص بهذه العملية يتم تمثيل ال leet كما يلي

Code(33)

```
use Acme::LeetSpeak;
$Who_we_Are="programing-freaks";
$string=leet($Who_we_Are);
print $string;
print "\n";
```

لاحظ في هذا الكود تم أستدعاء الموديل

Code(34)

```
use Acme::LeetSpeak;
```

الذي يكون مسؤول عن هذه العملية
ومن تم استعمال المتغير الذي يحمل قيمة

Code(35)

```
$Who_we_Are="programing-freaks";
```

على انها المعامل التابع لهذا المتغير ومن ثم تم استدعاء متغير اخر كان عمله ان يقوم بأستدعاء دالة ال

Code(36)

```
leet($Who_we_Are);
```

حيث في هذا الكود تم اعطاء خواص الدالة leet للمتغير $Who_we_Are
ألان لو نفذت هذا البرنامج سوف يكون ناتج تنفيذه كما يلي في الصورة التالية

```
use Acme::LeetSpeak;
$Who_we_Are="programing-freaks";
$string=leet($Who_we_Are);
print $string;
PR09ra/|/|I/\/9-|#R3/-\|<5
```

Figure(21)

هكذا تم تحويل كلمة ال programming-freaks من هذا النمط العادي الى النمط الخاص بال leets
ملاحظة
لاحظ الصورة التالية

```
use Acme::LeetSpeak;
$Who_we_Are="programing-freaks";
$string=leet($Who_we_Are);
print $string;
print "\n";
PROHgr/-\/\/\I|\|G-FRe@K$
```

Figure(22)

لقد تم أستعمال السلسلة النصية ذاتها في كل من البرنامجين ولكن الذي حصل هو اختلاف في ال output لكل من البرنامجين على الرغم من أنه تم استعمال السلاسل النصية ذاتها السبب في هذه العملية هو بسبب ما أحب أن اسميه (**<u>"بخواص الاستبدال"</u>**) الموجود في السورس الخاص بهذا الموديل اي بعبارة أوضح ما أقصده هو

```
'a', [ '4', '/-\\', '@', ],
'b', [ '8', '|3', '(3', ],
'c', [ '[', '<', '(', '{', ],
'd', [ ')', '[)', '|)', ],
'e', [ '3', ],
'f', [ '|=', '|#', ],
'g', [ '9', ],
'h', [ '#', '/-/', '[-]', ']-[', ')-(', '}{', '|-|' ],
'i', [ '1', '!', '|', '][', ],
'j', [ '_|', '_/', ],
'k', [ '|<', ],
'l', [ '1', '7', '1_', '|', '|_', ],
'm', [ '|\\/|', '/\\/\\', '/|/|', ],
'n', [ '|\\|', '/\\/', '[\\]', ],
'o', [ '()', 'oh', '0', ],
'p', [ '|o', '|*', '|>', ],
'q', [ '0_', '(,)', ],
'r', [ 'r' ],
's', [ '5', '$', ],
't', [ '7', '+', ],
'u', [ '(_)', '|_|', ],
'v', [ '\\/', '\\|', '|/', ],
'w', [ '\\/\\/', '\\^/', '\v/', ],
'x', [ '><', '}{', ],
'y', [ 'j', '@', '`/', ],
'z', [ 'z' ],
```

Figure(23)

لاحظ انه لكل حرف هنا يوجد له بديل وفي كل مرة تنفذ فيها الجملة يتم أما عرض الحرف بنفس ما تم عرضه سابقا او عرض بديل اخر لم يتم عرضه في التنفيذ السابق و هذا هو سبب حصول هذه العملية

How To Color Output

في هذا الموضوع ستتم مناقشة طرق تلوين النص في لغة البيرل تتم عملية تلوين المخرجات في لغة البيرل من خلال استعمال الموديل Term::Ansicolor
ويتم التمثيل البرمجي لهذه العملية كما يلي

Code(37)

```
use Term::ANSIColor;
  print color 'bold blue';
  print "This text is bold blue.\n";
  print color 'reset';
  print "This text is normal.\n";
  print colored ("programming-fr34ks.", 'yellow on_magenta'), "\n";
  print "This text is normal.\n";
  print colored ['yellow on_magenta'], 'Yellow on magenta.';
  print "\n";
```

هذا هو كود المسئول عن عملية التلوين في لغة البيرل لاحظ انه يحتوي على عدد من الخصائص وهي المقطع الاول

Code(38)

```
use Term::ANSIColor;                    #1
  print color 'bold blue';             #2
  print "This text is bold blue.\n";     #3
  print color 'reset';                 #4
```

1-
استدعاء الموديل المسئول عن تلوين النص

2-
الجملة الثانية من هذا المقطع هي انه يتم استدعاء الرويتن الخاص باللون bold blue وفي هذه الحالة تم اصبح اللون الازرق في حالة استعداد و stand by من اجل ان يتم تنفيذه

3-
هنا في الخطوة هذه تمت طباعة الجملة باللون الازرق وكان السبب في طباعة الجملة في اللون الازرق لانه في حالة الاستعداد

4-
هذه الخطوة هي خطوة مهمة جدا حيث ان عمل هذه الخطوة يكون في أعادة الخط في الشيل بعد التنفيذ الى ماكن عليه اي هنا لو لم تكن هذه الخطوة موجودة لاصبح كل اللون الشيل هو ازرق على الاستمرار حتى بعد انتهاء التنفيذ و الخروج من البرنامج لذا هذه الخطوة تلافت هذه المشكلة وعملت على لون الخط في الشيل الى ما كان عليه قبل التنفيذ

المقطع الثاني

**Code(39)*

```
   print "This text is normal.\n";                                     #1
print colored ("programming-fr34ks.", 'yellow on_magenta'), "\n";   #2
print "This text is normal.\n";                                      #3
print colored ['yellow on_magenta'], 'Yellow on magenta.';           #4
print "\n";
```

1-
هنا في الخطوة هذه يتم طباعة جملة نصية عادية
2-
هنا في الخطوة الثانية من هذا المقطع البرمجي ايضا تم استدعاء طريقة تلوين من أجل ان يتم تلوين كلمة او نص او سطر معين ولكن ما يهمنا من هذه الطريقة المختلفة عن الطريقة السابقة هو ان الموديل وفر لنا عناء استعمال خاصة ال reset مع هذا الاسلوب من التلوين و هكذا لن نحتاج الى هذه الطريقة مزيد من الخطوات اذا كان من من الممكن ان يتم استعمال اسلوب اسهل
4-
نفس الاسلوب في السطر 2 ولكن في style مختلف ولكن النتيجة هي نفس النتيجة
الان لاحظ صورة ناتج البرنامج بعد التنفيذ

```
 use Term::ANSIColor;
    print color 'bold blue';
    print "This text is bold blue.\n";
    print color 'reset';
    print "This text is normal.\n";
    print colored ("programming-fr34ks.", 'yellow on_magenta'), "\n";
    print "This text is normal.\n";
    print colored ['yellow on_magenta'], 'Yellow on magenta.';
    print "\n";
This text is bold blue.
This text is normal.
programming-fr34ks.
This text is normal.
Yellow on magenta.
```

Figure(24)

ملاحظة هامة عليك ان تفهمها هي انه يوجد فرق بين كل من

**Code(40)*

Output coloring	Output highlighting

الفرق هو انه في ال coloring التلوين يتم وفق ما أنت تريد بعبارة اوضح انت الذي تحدد ان يتم تلوين كلمة بيرل في اللون الاحمر وكلمة بايثون باللون الاصفر وكلمة لينكس بالاخضر ورقم 2 بالابيض ورقم 99 بالاسود

على سبيل المثال
ولكن لاحظ انه افترضنا في المثال انه البيرل بالاحمر و البايثون بالاصفر على الرغم من انه كل من الكلمتين هما strings ونفس الوضع مع الارقام
اما عن ال highlighting فانه يتم وفق قواعد محددة تكون كما يلي على سبيل المثال

strings =>red
numbers =>yellow
op =>black
func =>blue

```
use Term::ANSIColor;
   print color 'bold blue';
   print "This text is bold blue.\n";
   print color 'reset';
   print "This text is normal.\n";
   print colored ("programming-fr34ks.", 'yellow on_magenta'),
   print "This text is normal.\n";
   print colored ['yellow on_magenta'], 'Yellow on magenta.';
   print "\n";
```

Figure(25)

صورة عن ال highlighting وفق قواعد برنامج ال kwrite لبرنامج الذي ذكرناه قبل قليل

How To Encrypt

كيف تتم عملية تشفير السلاسل النصية في لغة البيرل هذه العملية تتم من خلال أستعمال الدالة الخاصة بعملية التشفير وهي دالة ال crypt واسلوب تمثيل هذه الدالة برمجيا هو كما يلي

Code(41)

```
$a="programming-fr34ks";
$b="PF";
$c=(crypt($a,$b));
print $c;
```

في هذا البرنامج الموجود في الاعلى تم أستعمال سلسلتين نصيتين ومع هذا فأن التشفير يبقى للمتغير الاول فقط ماهو سبب هذا ؟؟

فعلا ان التشفير يبقى لسلسلة نصية واحدة ولكن المتغير الثاني المستخدم في البرنامج يعتبر **"مفتاح تشفير"** اي لو لاحظت ناتج تنفيذ البرنامج فأن الحرفين الاولين من الناتج من تنفيذ البرنامج هما الحرفين pf وهذه هي صورة تنفيذ البرنامج

```
$a="programming-fr34ks";
$b="PF";
$c=(crypt($a,$b));
print $c;
PFqC/mbCtVS9c
```

Figure(26)

ملاحظات

اولا

مفتاح التشفير

المتغير الثاني الذي يستعمل في دالة التشفير كما ذكر انه يستخدم من أجل ان يكون مفتاح تشفير حيث يتم أخذ اول حرفين من هذا المتغير من أجل ان تكون مفتاح لتشفير المتغير الاول

ثانيا

دالة التشفير دالة مبرمجة على ان تكون قادرة على التعامل مع متغيرين أي تتعامل مع المتغير الاول الذي سوف يتم تشفيره و المتغير الثاني الذي سوف يتم اعتباره مفتاح التشفير واذا حاولت استعمال هذه الدالة من دون أحد هذين المتغيرين فأن الناتج الذي سوف تحصله هو الناتج الاتي

```
Not enough arguments for crypt at - line 2, near "$a)"
Execution of - aborted due to compilation errors.
```

Figure(27)

How To Chomp

دالة ال chomp هي من الدوال التي تتعامل مع السلاسل النصية في لغة البيرل و يكون تمثيلها البرمجي كما يلي

Code(42)

```
$a="programming-fr34ks\n";
$b=chomp($a);
print $b;
print "\n";
print $a;
```

لاحظ انه عمل هذه الدالة هو انه تعمل على ألغاء رمز الهروب"\n"

وتعمل على اعادته فيما لو اذا تم أسناده الى متغير لاحظ ناتج تنفيذ هذا البرنامج في الصورة المدرجة ادناه

```
$a="programming-fr34ks\n";
$b=chomp($a);
print $b;
print "\n";
print $a;
1
programming-fr34ks
```

Figure(28)

في هذا الصورة تلاحظ انه تم الغاء رمز الهروب واعادتة في متغير أخر

How To Chop

هذا النوع من الدوال ايضا يتم أستعماله مع السلاسل النصية في لغة البيرل اما عن تمثيله البرمجي فهو يكون كما يلي في هذا الكود

Code(43)

```
$a="Let`s Goo";
$b=chop($a);
print $a;
print "\n";
print "The retrned lette is $b";
```

الان لاحظ ما هو عمل هذه الدالة هي دالة يمكن أعتبارها دالة تعمل على ألغاء الحرف الاخير من السلسلة النصية و أعادته في متغير اخر لكي يعرض فيما لو أسند الى متغيرأخر وهذا المتغير الاخر لو تم طلبه من قبل المستخدم في عملية طباعة سوف يعطي المحتوى الموجود في داخله كما في الصورة المدرجة أدناه

```
$a="Let`s Goo";
$b=chop($a);
print $a;
print "\n";
print "The retrned lette is $b";
Let`s Go
The retrned lette is o
```

Figure(29)

How To Qw

هذه الدالة تعتبر من الدوال التنسيقية في لغة البيرل اي أن استعمالها مجرد استعمال من أجل أظهار او أضفاء لمسة على السلسلة النصية لاحظ هذا الكود البسيط

Code(44)

```
print "Hi,Iam perl programmer";
```

هذه الخطوة البرمجية بكل بساطة سوف تظهر لك ما يلي عند التنفيذ

```
print "Hi,Iam perl programmer";
Hi,Iam perl programmer
```

Figure(30)

ولكن ما يحصل عند أستخدام هذه الدالة هو ما يلي لاحظ الكود الاتي

Code(45)

```
print qw/"Hi,Iam perl programmer"/;
```

عند تنفيذ هذا الكود فان هذه الدالة تعمل على أعادة ال list الى سلسلة نصية لاحظ صورة تنفيذ البرنامج

```
print qw/"Hi,Iam perl programmer"/;
"Hi,Iamperlprogrammer"
```

Figure(31)

كما في ناتج ألتنفيذ يمكن معرفة انه عمل هذه الدالة بسيط وهو لا يتعدي مجرد أعادة الجملة المستخدمة الى سلسلة نصية ولا شئ أكثر من هذا

How To Equalize Between 2 Strings

الطريقة الاولى
لو كانت السلاسل النصية متساوية
كيف تتم معرفة هل انه السلاسل النصية المستخدمة في البرنامج هي سلاسل متساوية أم لا هذه العملية تتم من خلال استعمال هذه الدالة الخاصة بهذه العملية و هي دالة ال eq
وتتم العملية كما في السكرييت البرمجي التالي

Code(46)

```
$a="123456";
$b="123456";
if ($a eq "$b"){
print "good";
}
else {
print "bad";
}
```

الان لو تمت مقارنة هذه السلسلتين النصيتين مع بعضهما سوف تحصل على ناتج يؤكد لك انه كل من السلسلتين المستخدمتين في البرامج هي سلاسل متساوية لاحظ ناتج تنفيذ البرنامج كما في الصورة الاتية

```
$a="123456";
$b="123456";
if ($a eq "$b"){
print "good";
}
else {
print "bad";
}
good
```

Figure(32)

هذه الحالة لو كانت السلاسل النصية المستخدمة في البرنامج متساوية

الطريقة الثانية
عندما تكون السلاسل غير متساوية
عندما تكون السلاسل النصية المستخدمة في البرنامج غير متساوية يتم استعمال الدالة الخاصة بهذه العملية وهي دالة ال ne ويكون تمثيل هذه الدالة برمجيا من خلال هذا الكود الاتي

Code(47)

```
$a="PERL";
$b="perl";
if ($a ne $b){
print "these strings are not equal";
}
else {
print "these strings are equal";
}
```

الان لو تم تنفيذ هذا البرنامج سوف تحصل على القيمة الخاصة التي تدل على أنه السلاسل النصية المستخدمة في هذا البرنامج هي سلاسل نصية غير متساوية لاحظ الصورة الخاصة بتنفيذ البرنامج أدناه

```
$a="PERL";
$b="perl";
if ($a ne $b){
print "these strings are not equal";
}
else {
print "these strings are equal";
}
these strings are not equal
```

Figure(33)

ومن خلال هاتين الطريقتين تعلم فيما أذا كانت السلاسل النصية التي تستخدم في البرنامج هي سلاسل متساوية ام غير متساوية

How To Crunch

في هذا الموضوع سوف نتكلم عن موضوع التحكم في المسافات الموجودة في السلاسل النصية وكيف يتم الغاء المسافات البيضاء الغير مرغوب فيها من السلسلة النصية سواء كانت هذه الفراغات من نوع

1-skip sequence
2- trail white space

هذه العمليات التي تكلمنا عنها تتم من خلال استعمال الموديل الاتي

Code(48)

```
use String::Util;
```

والان لاحظ هذا الكود الذي تم دمج نوعين من المسافات التي قد تواجهها في أثناء كتابتك للبرنامج والمسافات المستعملة في هذا البرنامج هي نفس أنواع المسافات التي ذكرتها في الاعلى قبل قليل اليكم الكود الخاص بهذه العملية

Code(49)

```
use String::Util ':all';
$a="iam\tperl\tprogrammer";
$b="iam                perl     programmer";
$c=crunch($a);
$d=crunch($b);
print $c,"\n";
print $d,"\n";
```

الان لاحظ انه عندما يتم تنفيذ البرنامج فأن ناتج تنفيذه هو الناتج الأتي

```
use String::Util ':all';
$a="iam\tperl\tprogrammer";
$b="iam         perl     programmer";
$c=crunch($a);
$d=crunch($b);
print $c,"\n";
print $d,"\n";
iam perl programmer
iam perl programmer
```

Figure(34)

لاحظ الان عندما تم تنفيذ البرنامج قامت الدالة المستخدمة في هذا الموديل بألغاء المسافات والعمل على أعادة السلسلة النصية الى وضع اعتيادي بحيث انه لايوجد أكثر من مسافة واحد تفصل بين كلمة وكلمة أخرى

How To Nospace

اما الان سوف نتناول جزئية أخرى من الموديل الذي تكلمنا عنه وهذه الجزئية هي جزئية ال nospace واسلوب تمثيلها البرمجي هو كما يلي في هذا الكود

Code(50)

```
use String::Util ':all';
$a="We are perl programmer";
$b=nospace($a);
print $b;
```

الان لو تم تنفيذ هذا البرنامج سوف يكون ناتج تنفيذه هو كما يلي في الصورة الاتية

```
use String::Util ':all';
$a="We are perl programmer";
$b=nospace($a);
print $b;
Weareperlprogrammer
```

Figure(35)

عندما تم تنفيذ هذا البرنامج فأن الدالة المستعملة معه تعمل على اعادة السلسلة النصية التي تمت معاملتها بها على أن تطبع بدون ان تحتوي على فراغات بين الكلمات اي تطبع على أساس انها كلمة واحدة لا فواصل بينها

How To Hascontent

نبقى في نفس الموديل كيف تعرف ان المتغير الذي تعمل عليه و الذي يحمل سلسلة نصية هو متغير معرف ام لا ؟؟ هذا الموديل يوفر لك دالة وتقنية مفيدة تعرف من خلالها هل ان المتغير الذي تعمل عليه هو متغير ام معرف ام متغير معرف وهذه الطريقة لكي تمثل هذه الطريقة و برمجيا اليكم الكود

1-defined

*Code(51)

```
use String::Util ':all';
$a="perl programming";
if (hascontent ($a)){
print "\$a hascontent and it`s content is:->",$a;
}
else {
print "\$a is do`t have any content";
}
```

الان لو نفذت هذ البرنامج سوف تحصل على نتيجة وهذه النتيجة تخبرك بأن المتغير الذي تتعامل معه هو معرف وهذا ناتج تنفيذ البرنامج

```
use String::Util ':all';
$a="perl programming";
if (hascontent ($a)){
print "\$a hascontent and it`s content is:->",$a;
}
else {
print "\$a is do`t have any content";
}
$a hascontent and it`s content is:->perl programming
```

Figure(36)

2- undefined

الحالة الاولى التي تكلمنا عنها هي عندما يكون المتغير معرف والان سوف نتناول الحالة التي يكون فيها المتغير المستعمل في البرنامج هو متغير غير معرف وهذه هي الطريقة

***Code(52)**

```
use String::Util ':all';
undef $a;
if (hascontent ($a)){
print "the variable is defined";
}
else {
print "sorry the variable is not defined";
}
```

الان لو تم تنفيذ هذا البرنامج فأن الناتج من تنفيذه سوف يخبرك بأن المتغير الذي تتعامل معه هو متغير غير معرف واليكم الصورة الناتجة من تنفيذ البرنامج

```
use String::Util ':all';
undef $a;
if (hascontent ($a)){
print "the variable is defined";
}
else {
print "sorry the variable is not defined"
}
sorry the variable is not defined
```

Figure(37)

عمل هذه الدالة من الناحية البرمجية يمكن القول أنها مساوية للدالة ال built in والتي تحمل الاسم defined

How To Rand

أيضا هذا الموديل يوفر لك امكانية توليد كلمات عشوأئية لمساحات من الاحرف التي الذي تحددها وتتم عملية التوليد العشوأئي لاحرف كما يلي من خلال هذا الكود

Code(53)

```
use String::Util ':all';
$a = randword(5);
print $a;
```

الان لو تم تنفيذ البرنامج فأن الدالة المسئولة عن هذه العملية سوف تقوم بتوليد كلمة عشوأئية على طول حرفي مقدأره خمسة أحرف لا أكثر ولا أقل
اليكم ناتج تنفيذ البرنامج

```
use String::Util ':all';
$a = randword(5);
print $a;
rP6Gm
```

Figure(38)

How To Make String Banner

في هذا الموضوع سوف يكون الاتجاه البرمجي حول كيفية طباعة السلاسل النصية وكيفية التحكم بخصائص النص و اساليب طباعة النص بطرق مختلفة هذا النوع من العمليات البرمجية في لغة البيرل تتم من خلال استعمال موديل خاص لهذه العملية ويكون التمثيل البرمجي لهذه العملية كما يلي من خلال هذا السكريبت الاتي

Code(54)

```
use Text::Banner;
$a = Text::Banner->new;
$a->set('P3rL');
$a->size(1);
$a->fill('*');
$a->rotate('h');
print $a->get;
```

لاحظ البرنامج الموجود في الاعلى انه يحتوي على عدد كبير من ال methods من أجل ان يتم استعراض السلسلة النصية بالشكل المطلوب لاحظ الشكل التالي الذي يوضح ناتج تنفيذ هذا البرنامج

Figure(39)

لاحظ الخصائص الموجودة في البرنامج هي كما يلي

اولا

Code(55)

```
new
```

هي الخاصية التي تكون مسئولة عن تكوين ال object reference والرفرنس الذي سيكون فيما فيما بعد الذي يستخدم لتعريف السلسلة النصية

ثانيا

Code(56)

```
set
```

هذه الخاصية هي الخاصية التي تكون مسئولة عن تكوين الكلمة او السلسلة النصية التي سيتم التلاعب بها

ثالثا

Code(57)

```
size
```

هذه الخاصية هي الخاصية التي تكون مسئولة عن أعطأء الكلمة التي نود أن نستعرضها الحجم المرغوب

رابعا

Code(58)

```
fill
```

هذه الخاصية هي خاصية ملئ السلسلة النصية بالحرف او الرمز الذي نود ان تستعرض الكلمة به

خامسا

Code(59)

```
rotate
```

هذه الخاصية هي الخاصية التي تكون مسئولة عن موقع الكلمة من ناحية الاتجاه و هذه العلاقية تأتي بخيارين هما

rotate	meaning
h	أفقي
v	عمودي

ملاحظات هامة

اولا

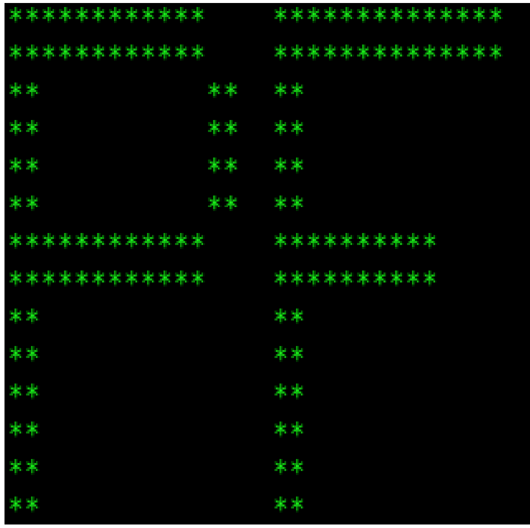
ان خاصية ال size هي الخاصية التي تكون مسئولة عن اعطاء الحجم للكلمة ولكن هي تعمل على استعمال الرمز المعطى في خاصية ال fill حيث تعمل تكراره بعدد المرات التي يكون موجود فيها في الخاصية size

لاحظ المقطع البرمجي الاتي

Code(60)

```
use Text::Banner;
$a=Text::Banner->new;
$a->set('PF');
$a->size(2);
$a->fill("*");
$a->rotate("h");
print $a->get;
```

والان لاحظ ناتج تنفيذ هذا البرنامج فأنه سيكون كما يلي فأن الشكل التالي يوضح كيف يكون ناتج تنفيذه



Figure(40)

لاحظ لانه ذكرنا في الخاصية الخاصة بالحجم ان الحجم المرغوب هو الحجم رقم (2) فأن الكلمة الناتجة من عملية من تنفيذ البرنامج مكونة من صفين من النجوم كما تم الطلب من البرنامج ان يفعل

How To Make Conversion

1- Arrays to strings

في هذا الموضوع سيكون الحديث عن تحويل أنواع المتغيرات الموجودة في لغة البيرل ألا وهي المصفوفات و الهاشات الى الى سلسلة نصية وهذه عملية مهمة للغاية عندما تكون في حاجة الى عملية تحويل بين انواع المتغيرات والكود البرمجي الاتي سوف يتناول تحويل المصفوفة الى سلسلة نصية

***Code(61)**

```
 use String::Escape qw( string2list list2string );
@list=("Storm","StrikerX","SpAwN","Snix","Dj");
 $list = list2string( @list );
 print $list,"\n";
```

الان لو تلاحظ ناتج تنفيذ هذا البرنامج فأنه سوف يكون تحويل المصفوفة التي تم أعطائها في البرنامج الى سلسلة نصية ويوضح الشكل الاتي ناتج تنفيذ هذا البرنامج اليكم الصورة

```
 use String::Escape qw( string2list list2string );
@list=("Storm","StrikerX","SpAwN","Snix","Dj");
 $list = list2string( @list );
 print $list,"\n";
Storm StrikerX SpAwN Snix Dj
```

Figure(41)

2- hashes to Strings

هذه الطريقة تستعمل في التحويل من المتغيرات التي تكون من نوع الهاش الى سلاسل نصية وهذه الطريقة تشبه الطريقة السابقة وهذا هو الكود المسؤول عن هذه العملية

***Code(62)**

```
 use String::Escape qw( hash2string );
%hash = (
	St0rM		=>"C Nightmare",
	Striker		=>"python Guru",
	Mutanti0n	=>"Ruby G33k",
	);
 $hash = hash2string( %hash );
 print $hash,"\n";
```

يوضح الشكل الاتي ناتج تنفيذ هذا الكود وكيف تم تحويل متغير من نوع هاش الى سلسلة نصية

```
  use String::Escape qw( hash2string );
%hash = (
        St0rM           =>"C Nightmare",
        Striker         =>"python Guru",
        Mutanti0n       =>"Ruby G33k",
        );
  $hash = hash2string( %hash );
  print $hash,"\n";
Mutanti0n="Ruby G33k" Striker="python Guru" St0rM="C Nightmare"
```

Figure(42)

P3rL ArrayS

Perl Arrays

النوع الثاني من المتغيرات في لغة البيرل هو المصفوفات التي تعتبر من المتغيرات المهمة و التي يتم استعمالها كثيرا في لغة البيرل ويكون التعريف البرمجي العام للمصفوفات كما يلي من خلال الكريبت البرمجي

Code(63)

```
@a=("Spawn","striker","Storm","Snix");
```

هذا هو التعريف العام للمصفوفات في لغة البيرل لاحظ دائما انه في كل مكان في اي سكريبت برمجي تلاحظ وجود علامة البريد الالكتروني @ عليك ان تعرف انه نوع المتغيرات البرمجية التي تتعامل معها هي المصوفوفات ولاحظ هنا في السطر البرمجي أعلاه ان الكود يحتوي على مصفوفة تحمل الاسم(a)
و هذه المصفوفة مكونة من اربعة عناصر كما هو مبين في هذا السطر البرمجي عليك ان تعرف هذه الملاحظة البرمجية المهمة هي ألمصفوفة في لغة البيرل تكون العناصر فيها صفرية الفهرسة اي انه العنصر الاول يكون مركزه التسلسلي هو صفر و ليس واحد

How to print array element

الان نأتي الى هذه الفقرة الخاصة بكيفية طباعة عناصر المصفوفة البرمجية تتم عملية طباعة عناصر المصفوفة في لغة البيرل كما يلي

Code(64)

```
@a=("Spawn","striker","Storm","Snix");
print $a[0];
```

الان لو تلاحظ ناتج تنفيذ هذا البرنامج كما في الصورة التالية سوف تفهم ما هو عمل عملية الطباعة في لغة البيرل

```
@a=("Spawn","striker","Storm","Snix");
print $a[0];
Spawn
```

Figure(43)

هذه الحالة اذا نفذت البرنامج وكان العنصر الذي ادخلته هو رقم صفر سوف يطبع اول اسم واذا ادخلت رقم واحد سوف يطبع العنصر الثاني ومن الممكن ان يتم طباعة اكثر من عنصر ومن الممكن ان يكون عناصر المصفوفة مكونة من أكثر نوع البيانات اي ارقام أو اسماء والخ من البيانات

Array special variable

في المصفوفات في لغة البيرل يوجد متغير يدعى هذا المتغير بالمتغير المميز وحقيقة هذا المتغير مميز فعلا نظرا لأهمية العمل الذي يقوم به اثناء برمجة المصفوفات ويكون اسلوب عمله البرمجي كما يلي في هذا الكود

Code(65)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
print $a[$#a];
```

لاحظ هذا السطر البرمجي لايحتوي على اسم العنصر الذي نريد ان نقوم بطباعته مثل الكود السابق الذي ذكرنا فيه ان العنصر الذي نريد ان نقوم بطباعته وهذا هو سبب تميز هذا المتغير الان لو تم تنفيذ هذا الكود فأن الناتج من عملية التنفيذ هذه سوف تكون كما يلي في الصورة الاتية

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n
");
print $a[$#a];
mutanti0n
```

Figure(44)

لاحظ انه عند تم تنفيذ هذا البرنامج فأن الناتج من عملية التنفيذ هذه انه يتم طباعة العنصر الاخير من عناصر المصفوفة اي أنه عمل هذا المتغير الخاص هو انه يعمل على طباعة العنصر الاخير الموجود في المصفوفة مهما كان عدد العناصر المكونة للمصفوفة كثيرا او قليلا

How print the all array

لاحظ ماذا لو كانت لديك مصفوفة كالمصفوفة التي تم ذكرها في الاعلى وقمت بطباعة المصفوفة كاملة ماذا سيكون ناتج تنتفيذ هذه العملية لاحظ هذا الكود البرمجي

Code(66)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
print @a,"\n";
```

لو نفذت الكود البرمجي اعلاه سيكون ناتج تنفيذ هذا البرنامج هو ما يلي في الصورة الاتية

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n
");
print @a,"\n";
spawnStrikerStormDjSnixbalckraymutanti0n
```

Figure(45)

ناتج تنفيذ هذه العملية هو طباعة جميع عناصر المصفوفة كما وردت في الكود من دون وجود فواصل او فراغات بين عناصر المصفوفة وهنالك من يظن ان السبب في هذه العملية هو عدو وجود فراغات بين عناصر المصفوفة

في البرنامج الرئيسي ولكن لاحظ هذا السكريبت البرمجي

Code(67)

```
@a=("spawn" , "Striker" , "Storm" , "Dj" , "Snix");
print @a,"\n";
```

لو نفذت البرنامج المذكور أعلاه سوف تكون نتيجة التنفيذ هي النتيجة الواضحة في الصورة أدناه

```
@a=("spawn" , "Striker" , "Storm" , "Dj" , "Snix");
print @a,"\n";
spawnStrikerStormDjSnix
```

Figure(46)

ناتج تنفيذ هذا الكود هو نفس ناتج تنفيذ البرنامج السابق اي أنه حتى لو تم وضع فراغات بين عناصر المصفوفة فأنه عند عملية الطباعة هذه الفراغات سوف تهمل ولن تؤثر على ناتج تنفيذ البرنامج

هذه الحالة من الممكن ان يتم أعتبارها مشكلة لو كانت لديك مصفوفة كبيرة ومكونة من عدد من عناصر كبير لذا من الممكن ان يتم التخلص من هذه المشكلة باستخدام هذه الطريقة

*code(68)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
print "@a","\n";
```

لاحظ انها طريقة سهلة من شأنها أن تعمل على حل هذه المشكلة أنظر الى ناتج تنفيذ هذا الكود في الصورة الاتية

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n
");
print "@a","\n";
spawn Striker Storm Dj Snix balckray mutanti0n
```

Figure(47)

How to undef the array

من الممكن ان تقوم بألغاء تعريف عنصر من عناصر المصوفة وبذلك يتم الغاء هذه العنصر من هيكلة المصفوفة اي يصبح عنصر غير معرف و بالتالي يصبح عنصر غير موجود الطريقة البرمجية التي تتم بها هذه الطريقة هي

Code(69)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
undef $a[0];
print "@a","\n";
```

الان لو تم تنفيذ هذا البرنامج سوف يكون ناتج تنفيذ هذا البرنامج هو الناتج الأتي في الصورة ادناه

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n
");
undef $a[0];
print "@a","\n";
 Striker Storm Dj Snix balckray mutanti0n
```

Figure(48)

من خلال استعمال الدلة ال undef تم ألغاء تعريف العنصر الاول من المصفوفة وهو العنصر الذي يحمل الاسم spawn ومن ثم عندما تم طباعة المصفوفة تم الغاء هذا العنصر من هيكلة المصفوفة

هذه العملية من الممكن ان يتم استعمالها على أكثر من عنصر في المصفوفة ولكن بالطريقة الاتية

الطريقة الاولى

Code(70)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
undef $a[0],$a[2];
print "@a","\n";
```

هذه الطريقة هي طريقة خاطئة لانه في هذه الحالة العنصر الاول من المصفوفة سوف يتم الغاءه اما العنصر الثالث فلن يتم الغاءه لانه عمل هذه الدالة يقتصر على عنصر واحد فقط

الطريقة الثانية

Code(71)

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n");
undef $a[0];
undef $a[2];
print "@a","\n";
```

هذه الطريقة هي الطريقة الصحيحة من أجل الغاء تعريف أكثر من عنصر في المصفوفة وألان لاحظ ناتج تنفيذ هذا الكود في الصورة الاتية

```
@a=("spawn","Striker","Storm","Dj","Snix","balckray","mutanti0n
");
undef $a[0];
undef $a[2];
print "@a","\n";
 Striker  Dj Snix balckray mutanti0n
```

Figure(49)

ومن الممكن أيضا أن يتم استعمال دالة ال undef مع المصفوفات كاملة وليس مع عناصر المصفوفات فقط وهذه الطريقة الخاصة باستعمال هذه الدالة مع المصفوفات في الكود الاتي

Code(72)

```
@a=("ruby","python","perl","C");
undef @a;
print @a;
```

وناتج تنفيذ هذا البرنامج هو ان يتم الغاء المصفوفة كاملة وهذه هي صورة ناتج تنفيذ البرنامج

```
@a=("ruby","python","perl","C");
undef @a;
print @a;
```

Figure(50)

How to defined

هذه الدالة تستعمل لمعرفة هل انه المتغير الذي تتعامل معه هو متغير معرف ام لا. وهذه الدالة يتم استعمالها مع عناصر المصفوفة ويتم استعمالها مع المصفوفة كاملة
وهذه هي طرق استعمال هذه الدالة مع المصفوفات وعناصرها

Code(73)

```
@a=("Storm","striker","Dj","SniX","Spawn","Ray");
if (defined(@a)){
print "the array \@a is defined","\n";
}
```

في هذا المقطع البرمجي تم استعمال جملة اذا الشرطية مع دالة ال defined حيث تم لمعرفة اذا كانت المصفوفة معرفة ولاحظ اذا تم تحقيق الشرط طلبنا من جملة اذا ان تعمل الاتي

Code(47)

```
print "the array \@a is defined","\n";
```

ولكن عليك ان تلاحظ شئ مهم جدا هو انه في هذه الجزئية البرمجية من جملة الطباعة(\@a)تعني ان المصفوفة سوف تعامل على انها سلسلة نصية عادية وليس لها علاقة بالرفرنس من قريب او بعيد

How to iterate over the array

جمل التكرار في لغة البيرل تقسم الى قسمين هما

1-for
2-foreach

القسم الاول هو القسم المعروف لدى اغلب المبرمجين
القسم الثاني هو القسم الذي نحن بصدد مناقشته وهو القسم الذي يستعمل غالبا في برمجة المصفوفات
وطريقة تمثيل هذا الاسلوب البرمجي يكون كما يلي

Code(75)

```
@a=("blackray","Striker","Storm","mutanti0n","Snix");
foreach $a(@a){
print $a,"\n";
}
```

الان لو تم تنفيذ هذا الكود فان الناتج الذي سوف يظهر من عملية التنفيذ سوف يكون كما يلي في الصورة الاتية



Figure(51)

How to do math with foreach

من الممكن ان تقوم ببعض العمليات الرياضية في جملة التكرار foreach يعني من الممكن ان تتم عملية الجمع و والطرح
الاول علمية الجمع

Code(76)

```
@a=("1","2","3","4");
foreach $a(@a){
$a++;
print $a;
}
```

الان لو تم تنفيذ هذا الكود سوف يتم ملاحظة يتم زيادة عناصر المصفوفة بمقدار واحد وهذه الصورة الناتجة من عملية التنفيذ

```
@a=("1","2","3","4");
foreach $a(@a){
$a++;
print $a;
}
2345
```

***F**igure(52)*

هذا ما كان يتعلق بعملية الجمع
ثانيا عملية الطرح
اما ما يخص عمليات الطرح فأنه من الممكن ان تتم تمثيلها برمجيا كما يلي من خلال هذا الكود الاتي

```
@a=("1","2","3","4");
foreach $a(@a){
$a--;
print $a;
}
0123
```

***F**igure(53)*

How to make foreach for a Referenced array

من خلال استعمال جملة التكرار foreach من الممكن ان تتمكن من طباعة العناصر للمصفوفة التي تم عمل reference وتتم العملية برمجيا كما يلي

Code(77)

```
@a=("spawn","storm","striker","mutanti0n");
$a_ref=\@a;
foreach $a(@$a_ref){
print "$a are the fr34k programmers of programming-fr34ks","\n";
}
```

الان لو تم تنفيذ هذا البرنامج فأن ناتج التنفيذ منه سوف يكون كما يلي في الصورة الاتية

```
@a=("spawn","storm","striker","mutanti0n");
$a_ref=\@a;
foreach $a(@$a_ref){
print "$a are programmers of programming-fr34ks","\n";
}
spawn are programmers of programming-fr34ks
storm are programmers of programming-fr34ks
striker are programmers of programming-fr34ks
mutanti0n are programmers of programming-fr34ks
```

Figure(54)

الان ستتناول الخطوات الاتية الفقرات البرمجيةالمبهمة نوعا ما

الفقرة الاولى

Code(78)

```
$a_ref=\@a;
```

في هذه الفقرة البرمجية تم أسناد هذه المصفوفة التي تحمل الاسم a الى متغير اخر يحمل الاسم $a_ref تذكر دائما انه كل متغير مهما كان

1-scalar
2-array
3-hash

فأن هذا يعني انه المتغير الموجود في هذا المقطع البرمجي انه متغير تم أسناده الى رفرنس

الفقرة الثانية

Code(79)

```
foreach $a(@$a_ref){
```

في هذه الفقرة البرمجية تم أستعمال جملة التكرار لمتغير من داخل المصفوفة على المتغير الذي تم أسناده لكي يكون رفرنس او مصدر للمصفوفة التي تم أسناده اليه

How to sort

الان في هذا الموضوع سوف يكون المحور البرمجي حول الترتيب في داخل المصفوفة كيف يتم كيف يكون والتمثيل البرمجي لمثل هذه الحالات هو كما يلي

***Code(80)**

```
@a=("striker","storm","dj","snix","mutanti0n");
print "@a";
print "\n";
print sort(@a);
```

الان لاحظ كيف سيكون ناتج تنفيذ البرنامج مع الملاحظة انه البرنامج الموجود في الاعلى يحتوي على جملتي طباعة اليكم صورة التنفيذ

```
@a=("striker","storm","dj","snix","mutanti0n");
print "@a";
print "\n";
print sort(@a);
striker storm dj snix mutanti0n
djmutanti0nsnixstormstriker
```

Figure(55)

من النظرة الاولى الى صورة البرنامج من الممكن ان يتم ملاحظة الفرق بين جملتي الطباعة ولكن هنالك خاصية أخرى في دالة ال sort وهي كما يلي في هذه الفقرة البرمجية

```
@a=("Spawn","ahemd");
print (sort(@a));
Spawnahemd[spawn@loca
```

Figure(56)

لاحظ انه في هذا المثال يوجد كلمتين هما Spawnو ahmed وعلى الرغم من أنه الحرف الاول منها يأتي في قبل الحرف الاول الموجود في كلمة سباون ولكن عن التنفيذ تأتي كلمة سباون قبل كلمة احمد؟؟
السبب في هذه العملية انه ال letter case تختلف حيث في هذه الدالة تكون الحروف الكبيرة تكون لها الاولوية في في التنفيذ اي

upper case comes b4 the lower case

How to enumerate the elements

طبعا من المعلوم انه المصفوفة في لغة البيرل مكونة من عدد من العناصر واذا كان عدد العناصر فيها كبير اي انه اكبر من ان يتم عده فكريا او بواسطة اليد فأنه في هذه الحالة لغة البيرل قد وفرت دالة تقوم بهذه العملية وهي دالة ال scalar ويكون اسلوب تمثيلها البرمجي العام كما يلي من خلال هذا الكود

Code(81)

```
@a=("dj","Spawn","Snix","st0rm","Striker","pf","SG","google","balckray","mutati0n","1","2","3","4","5");
print (scalar(@a));
```

الان لو تم تنفيذ هذا الكود الموجود في الاعلى فأن ناتج تنفيذه هو كما يلي في الصورة الاتية



Figure(57)

How to pop

سبق و ان تم في الصفحات القليلة السابقة انه اذا اردت انه ان تعرف ما هو العنصر الاخير من المصفوفة التي تقوم ببرمجتها هنالك متغير خاص يقوم بهذه المهمة وهو المتغير $#ARRAYNAME

أما اذا لم تكن من محبي هذه الطريقة فأنه توجد دالة مبنية داخليا في لغة البيرل تقوم بهذه العملية وهي الدالة pop أما اسلوب تمثيل هذه الدالة برمجيا فأنه يكون كما يلي في هذا الكود

Code(82)

```
@a=("perl","c","php","python","ruby");
print pop(@a);
```

الان لو تم تنفيذ هذا الكود فأنه الناتج منه يكون كما يلي في الصورة الاتية

```
@a=("perl","c","php","python","ruby")
print pop(@a);
ruby
```

Figure(58)

الان لاحظ عندما تم تنفيذ هذا الكود فأنه الذي سيحدث في الكود هو فعلا ما يحدث عندما يتم استعمال المتغير الخاص $#ARRAYNAMEفي الحالتين يتم أعادة المتغير الاخير الذي يكون موجود في المصفوفة وفي كل الحالتين حجم المصفوفة غير مهم سواء كان كبير او صغير

How to push

كيف من الممكن ان تتم عملية اضافة عناصر للمصفوفة بعد ان تتم كتابة المصفوفة برمجيا يعني لو كانت المصفوفة لديك مكونة من ثلاثة عناصر وبعد ان قمت بكتابتها اردت ان تضيف عنصر اخر فأنه سيكون ضرب من ضروب المستحيل ان تعمل على كتابة البرنامج مرة اخرى من جديد فقط لكي تضيف عنصر أخر الى المصفوفة ولهذا فأن البيرل قد وفرت دالة تعمل على حل هكذا نوع المشاكل التي قد يقع بها بعض المبرمجين وهذه الدالة هي دالة ال push ويكون اسلوب تمثيلها البرمجي كما يلي في هذا الكودة

Code(83)

```
@a=("St0rm","Striker","Snix");
print "@a";
print "\n";
push (@a,"SpAwN");
print "@a";
```

الان لاحظ لم تنفيذ الكود الموجود في الاعلى فأن الناتج من عملية تنفيذه هو سوف يكون كما يلي

```
@a=("St0rm","Striker","Snix");
print "@a";
print "\n";
push (@a,"SpAwN");
print "@a";
St0rm Striker Snix
St0rm Striker Snix SpAwN
```

Figure(59)

لاحظ الفرق في ما بين الحالتين حيث جملة الطباعة الاولى قامت بطباعة العناصر على انهم 3 عناصر قبل ان يتم ان استعمال دالة دفع العنصر الجديد وعندما تم استعمالها وتم طباعة المصفوفة مرة اخرى تمت أضافة العنصر الذي اردنا ان نقوم بعملية اضافته

How to add commas

الان لو كتبت اي مصفوفة ولتكن هذه المصفوفة في الكود الاتي

Code(84)

```
@a=("dj","Spawn","Snix","st0rm","Striker","pf","SG","google","balckray","mutati0n","1","2","3","4
","5");
```

وقمت بطباعة هذه المصفوفة فأن الناتج من طباعتها هو ألاتي في الصورة الاتية

```
djSpawnSnixst0rmStrikerpfSGgooglebalckraymutati0n12345[
```

Figure(60)

فأنه سيكون من غير الممكن والصعب جدا ان تقرأ عناصر مصفوفة تمت طباعتها هكذا وجميع العناصر متداخلة مع بعضها لذا فأنه لابد من حل لهكذا نوع المشاكل وعند هذه النقطة فأن لغة البيرل تزودك بدالة مهمة جدا هي دالة الربط او ما تعرف بأسم دالة ال join

ويكون التمثيل البرمجي العام لهذا النوع من الدوال كما يلي في هذا الكود

Code(85)

```
@a=("dj","Spawn","Snix","st0rm","Striker","pf","SG","google","balckray","mutati0n","1","2","3","4
","5");
print  (join (",",@a));
print "\n";
# or u can use this way if u want
@a=("dj","Spawn","Snix","st0rm","Striker","pf","SG","google","balckray","mutati0n","1","2","3","4
","5");
print  (join ("-",@a));
```

الان لو تم تنفيذ البرنامج الموجود في الاعلى فأن الناتج من عملية تنفيذه يكون كما يلي في الصورة الاتية التي توضح عمل الدالة join

```
dj,Spawn,Snix,st0rm,Striker,pf,SG,google,balckray,mutati0n,1,2,
3,4,5
dj-Spawn-Snix-st0rm-Striker-pf-SG-google-balckray-mutati0n-1-2-
3-4-5
```

Figure(61)

How to reverse

الان سوف نتكلم عن الطريقة التي يتم استعمالها من أجل ان يتم عكس عناصر المصفوفة وهذه الطريقة تتم من خلال استعمال الدالة الخاصة بهذه العملية وهي دالة ال reverse ويكون تمثيلها البرمجي كما يلي من خلال هذا الكود الاتي

Code(86)

```
@a=("C","perl","python");
print (reverse(@a));
```

الان لو تم تنفيذ هذا الكود فأن الناتج منه يكون كما يلي هذه الصورة الاتية حيث تلاحظ انه سوف يتم عكس ترتيب العناصر عند التنفيذ اليكم الصورة

```
@a=("C","perl","python");
print (reverse(@a));
pythonperlC
```

Figure(62)

هذه هي الطريقة الاولى التي يتم استعمالها مع دالة العكس وهذا النوع الاول من طريقة العكس احب اطلق عليها أسم the ordered reversing

حيث هنالك طريقة اخرى تستعمل لعكس محتوى المصفوفة ويتم تمثيل هذه الطريقة كما يلي من خلال هذا الكود الاتي

Code(87)

```
@a=("python","C","perl");
$rev=reverse(@a);
print $rev;
```

الان لو تم تنفيذ هذا الكود الاتي فأن الناتج عملية التنفيذ هي كما يلي في هذا الكود الاتي

```
@a=("python","C","perl");
$rev=reverse(@a);
print $rev;
lrepCnohtyp
```

Figure(63)

تلاحظ تنفيذ الطريقة البرمجية الثانية الخاص بدالة ال reverse فأنها تعمل على عكس الموقع اي الموقع الاخير الاول والحرف الاخير من الكلمة التي ستعكس يصبح الحرف الاول ولكن هذا النوع من الاستعمال غير دارج كثيرا في استعمالات دالة العكس حيث الاستعمال الرسمي لهذه الدالة يكون في الطريقة الاولى

How to shift

هذه الدالة تعمل على عمل shift للعنصر الاول من المصفوفة ومن ثم تعمل على أعادته ومن ثم في هذه الحالة يتم تقليص عدد العناصر الموجودة في المصفوفة بمقدار واحد اي يكون اسلوب البرمجي الخاص بها يكون كما يلي في هذا الكود البرمجي

Code(88)

```
@a=("storm","spawn","Striker","Snix");
print (shift(@a));
print "\n";
print @a;
```

الان عندما يتم تنفيذ هذا البرنامج لاحظ كيف يتم اعادة العنصر الاول من المصفوفة وايضا انقاص عدد عناصر المصفوفة بمقدار واحد وهذه الصورة الناتجة من تنفيذ البرنامج

```
@a=("storm","spawn","Striker","Snix");
print (shift(@a));
print "\n";
print @a;
storm
spawnStrikerSnix
```

Figure(64)

How to delete

في هذا الموضوع سوف يكون المحور البرمجي عن كيفية التحكم في عدد عناصر المصفوفة وكيف يتم انقاص او مسح العناصر الموجودة داخل المصفوفة وتتم هذه العملية عن طريق استعمال الدالة ال delete
ويكون التمثيل البرمجي لهذه الدالة هو كما يلي في هذا الكود

Code(89)

```
@a=("Python","C","perl","ruby");
delete($a[2]);
print "@a","\n";
```

الان لو تم تنفيذ هذا الكود فأن ناتج تنفيذه هو كما يلي في الصورة الاتية في الشكل أدناه

```
@a=("Python","C","perl","ruby");
delete($a[2]);
print "@a",$x,"\n";
Python C  ruby
```

Figure(65)

الان لاحظ الى الشكل تلاحظ انه عند طباعة المصفوفة تم الغاء العنصر الثالث الذي كان موجود في المصفوفة قبل ان يتم استعمال دالة المسح معها

How to grep

على الرغم من ان هذه الدالة يكون أغلب استعمالها البرمجي مع خواص ما يتعلق ببرمجة ال regexpالأ انه بالرغم من هذا ايضا تستعمل في برمجة المصفوفات وايضا عن استعمالها مع المصفوفات تبقى الخواص التي تنطبق عليها في برمجة التعابير القياسية تبقى قيد التنفيذ في برمجة المصفوفات الان تمثيلها البرمجي العام يكون كما يلي في الكود البرمجي الاتي

Code(90)

```
@a=("spawn","Storm","Striker","snix");
print (grep /^S/,@a);
```

الان لاحظ انه لو تم تنفيذ هذا البرماج فأن الناتج من عملية التنفيذ هو كما يلي في الصورة الاتية

```
@a=("spawn","Storm","Striker","snix");
print (grep /^S/,@a);
StormStriker
```

Figure(66)

ناتج تنفيذ هذا الكود الاتي هو انه يقوم بطباعة الناتج الذي في الاعلى والسبب في أنه لم يطبع كل الكلمات التي تبدأ بحرف ال s لانه تم تحديد الحرف s في حالته الكبيرة الان لو أردت ان تطبع كافة عناصر المصفوفة من دون اخذ اي اعتبار لحالة الاحرف فعليك ان تضيف الاتي الى الكود

Code(91)

```
@a=("spawn","Storm","Striker","snix");
print (grep /^S/i,@a);
```

الان ناتج تنفيذ هذا الكود هو كما يلي في هذه الصورة الاتية

```
@a=("spawn","Storm","Striker","snix");
print (grep /^S/i,@a);
spawnStormStrikersnix
```

Figure(67)

الان لاحظ عندما استعملنا المعرف i الذي يعمل على الغاء حالة الاحرف في حالة التعابير القاسية تم طباعة كل العناصر الموجودة في المصفوفة

الجدول الاتي سيوضح الاختصارات والمعرفات الموجودة في الكود

الشرح	الايعاز	ت
اهمال حالة الاحرف التي تكون موجودة في المصفوفة اي لايهتم اذا كانت الاحرف كبيرة او كانت صغيرة	i	1
هذا المعرف يعني عند المطابقة تتم من بداية الكلمة	^	2
هذا المعرف يعني عند المطابقة يجب ان تتم هذه المطابقة من نهاية الكلمة	$	3

وفيما يتعلق في هذا الجدول سيتم شرح كل هذه المعرفات و ال anchors في كتاب خاص سيتناول برمجة التعابير القياسية في لغة البيرل حيث سيتم شرح هذه الامور بالتفصيل

How to unshift

هذه الدالة لاتعمل كما تعمل دالة ال shift كما يظن البعض ولكن هذه الدالة تعتبر دالة معاكسة لعمل دالة ال push اذ ان دالة ال push تعمل على اضافة العناصر الموجودة فيها الىاخر المصفوفة أما دالة ال unshift فأن اسلوب تمثيلها وعملها البرمجي يكون كما يلي في الكود اللاتي

Code(92)

```
@a=("spawn","perl","Storm","C");
$add=unshift(@a,"Striker","python");
print @a;
```

الان لاحظ تم تنفيذ هذا البرنامج الموجود في الاعلى فأن الناتج سيكون كما في الصورة الاتية

```
@a=("spawn","perl","Storm","C");
$add=unshift(@a,"Striker","python");
print @a;
print "\n";
StrikerpythonspawnperlStormC
```

Figure(68)

لاحظ الان انه عندما تم تنفيذ هذا الكود تم اضافة العناصر الموجودة في جملة البرمجة الخاصة بال unshift الى بداية المصفوفة وعلى عكس ما تقوم به دالة ال push
ولهذا يتم اعتبار هذه الدالة هي الدالة المعاكسة لدالة ال push

How to shuffle

الان لو كنات لديك مصفوفة واردت ان تولد عناصر عشوائية من هذه المصفوفة ففي هذه الحالة عليك ان تلجأ الى الموديلات الجاهزة الموجودة على شبكة ال Cpan لذا في هذه الحالة سوف نلجأ الى استعمال موديل ال list::Util

Code(93)

```
use List::Util qw(shuffle);
@array=("spawn","snix","storm","Striker","dj");
@array = shuffle(@array);
print @array;
```

الان لو تم تنفيذ هذا السكرييت البرمجي فأن الناتج من تنفيذ هذا السكرييت البرمجي كما يلي في هذه الصورة الاتية

```
use List::Util qw(shuffle);
@array=("spawn","snix","storm","Striker","dj");
@array = shuffle(@array);
print @array;
djstormStrikerspawnsnix
```

Figure(69)

الان تلاحظ عندما تم تنفيذ هذا السطر البرمجي فأن ناتج تنفيذه يكون عشوائي الخروج وفي كل مرة يتم تنفيذ هذا السكرييت البرمجي يتم طباعة المصفوفة بشكل عشوائي مختلف عن الاخر

وهكذا يكون اسلوب عملها

How to lock array

في هذا الموضوع سوف نتكلم عن كيفية قفل المصفوفة عن طريق استعمال موديل خاص يقوم بهذه العملية أي انه تصبح المصفوفة صالحة للقراءة فقط اي لا تكون تملك القدرة على اضافة اليها اي شئ من عناصر او تعديلات وتتم هذه العملية كما يلي من خلال هذا الكود الاتي

Code(94)

```
use Array::Lock qw(lock_array);
@array=("Spawn","perl");
lock_array(@array);
push(@array,"programming-fr34ks");
print @array;
print "\n";
```

الان تلاحظ من سياق البرنامج العام انه تم قفل المصفوفة وعلى الرغم من انه استعملنا دالة الدفع لكي نضيف عنصر جديد الى المصفوفة ولكن لاحظ ما الذي يجري عندما تريد القيام بهذا الصورة الاتية توضح ما الذي سوف يجري

```
use Array::Lock qw(lock_array);
@array=("Spawn","perl");
lock_array(@array);
push(@array,"programming-fr34ks");
print @array;
print "\n";
Modification of a read-only value attempted at - line 4.
```

Figure(70)

تلاحظ من الصورة الاتية انه التعريف الخاص بدالة ال push لن يتم لانه الدالة حاليا مقفلة وتسمح لك بالقراءة فقط في برمجة المصفوفات العادية من الممكن ان تغير قيمة عنصر من حالة الى حالة اخرى اليكم هذا الكود البرمجي الذي سوف يوضح مالذي أعنيه

*code(95)

```
@a=("programming-freaks");
$a[0]="programming-fr34ks";
print @a;
print "\n";
```

الصورة الاتية سوف توضح ناتج تنفيذ هذا البرنامج

```
programming-fr34ks
```

Figure(71)

وهذا يعني ان البيرل تسمح لك بأن تقوم بتبديل برمجة العناصر الموجودة داخل المصفوفة ولكن مع هذا الموديل الذي نتعامل معه هذه العملية تعتبر غير مسموحة لانه المصفوفة فقط للقراءة اي انه من غير الممكن ان تقوم ب بتبديل القيم التي تحملها العناصر الكود الاتي سوف يوضح الطريقة بشكل افضل

Code(96)

```
use Array::Lock qw (lock_array);
@a=("programming-freaks");
lock_array (@a);
$a[0]="programming-fr34ks";
print @a;
```

الان لاحظ انه لو تم تنفيذ هذا البرنامج فأنك سوف تحصل على رسالة تشبه تماما الرسالة التي كنت قد حصلت عليها من البرنامج السابق وهي الرسالة الاتية التي تفيد بانه المصفوفة للقراءة فقط

```
Modification of a read-only value attempted at - line 4.
```

Figure(72)

How to unlock

من الممكن ان يتم ابطال هذه الخاصية من خلال استعمال دالة اخرى من داخل هذا الموديل وهي دالة ال unlock_array والكود الاتي يوضح كيفية تمثيل هذه الطريقة

Code(97)

```
use Array::Lock qw (unlock_array);
@a=("programming-freaks");
unlock_array (@a);
$a[0]="programming-fr34ks";
print @a;
```

الان لو تقوم بتنفيذ هذا الكود سوف تلاحظ انه سوف يسمح لك بالقيام باستبدال القيم و الصورة الاتية توضح كيفية أتمام هذه العملية وكيف اصبح من الممكن ان يتم تغيير عناصر المصفوفة

```
use Array::Lock qw (unlock_array);
@a=("programming-freaks");
unlock_array (@a);
$a[0]="programming-fr34ks";
print @a;
programming-fr34ks
```

Figure(73)

The Array Utils

الان في هذا الموضوع سوف نناقش اساليب خاصة عن كيفية التلاعب بالمصفوفات الان سوف يكون النقاش عن موضوع وهو عن كيفية ايجاد العناصر المختلفة الموجودة في مصفوفتين أي ان يتم عمل مقارنة ويتم نبذ المتشابه بين المصفوفتين وطباعة المختلف وكما يلي من خلال هذا الكود الذي سوف يوضح الاسلوب البرمجي العام لهذه الطريقة

Code(98)

```
use Array::Utils qw(:all);
@a =("spawn","perl","programmer");
@b =("St0rm","C","programmer");
@diff = array_diff(@a, @b);
print "@diff","\n";
```

الان لاحظ انه لو تم تنفيذ هذا البرنامج فأن ناتج تنفيذه سوف يكون كما يلي في الصورة الاتية

```
use Array::Utils qw(:all);
@a =("spawn","perl","programmer");
@b =("St0rm","C","programmer");
@diff = array_diff(@a, @b);
print "@diff","\n";
spawn perl St0rm C
```

Figure(74)

تلاحظ كناتج للتنفيذ فانه يتن حذف اهمال كلمة programmer لانه كلمة موجودة في كل المصفوفتين ولكن كل من هذه الكلمات spawnperlst0rmc لم يتم اهمالها لانه كلمات غير مشتركة بين المصفوفتين تجدر الاشارة الى فقرة مهمة هذه الفقرة من الممكن ان يتم تطبيقها على مصفوفتين فقط اي لايجوز ان يتم تطبيقها على أكثر من مصفوفتين

How to intersect

الموضوع الذي تم تناوله في الاعلى والذي كان يشرح كيف يتم نبذ العناصر المتشابهة واخذ او طباعة العناصر الغير متشابهة ولكن هنا في هذا الموضوع سوف نعمل العكس تماما سوف نقوم بطباعة العناصر المتشابهة وترك او نبذ العناصر المختلفة وتتم هذه الطريقة عن طريق نفس الموديل الذي تم استعماله في الاعلى ولكن من خلال دالة أخرى اما عن التمثيل البرمجي الخاص بهذه العملية يتم كما يلي من خلال هذا الكود

Code(99)

```
use Array::Utils qw(:all);
@a=("mutanti0n","ruby","programmer","pf member");
@b=("Striker","python","programmer","pf member");
my @c = intersect(@a, @b);
print "@c","\n";
```

أما عن ناتج تنفيذ البرنامج فانه موضح كما في الشكل التالي

```
use Array::Utils qw(:all);
@a=("mutanti0n","ruby","programmer","pf member");
@b=("Striker","python","programmer","pf member");
my @c = intersect(@a, @b);
print "@c","\n";
programmer pf member
```

Figure(75)

لاحظ في الصورة كيف تم أعتماد العناصر المتشابهة ونبذ المختلفة

How to unique

هذا المضوع ايضا يكون بخصوص برمجة العناصر وترك المتشابهة واعتماد المختلفة هو يشبه الموضوع الذي تم ذكره قبل قليل ولكن فيه لعض الاختلافات على كل الموديل الذي يكون مسؤول عن هذه العملية هو array::unique ويكون التمثيل البرمجي العام لهذا الموديل كما يلي

Code(100)

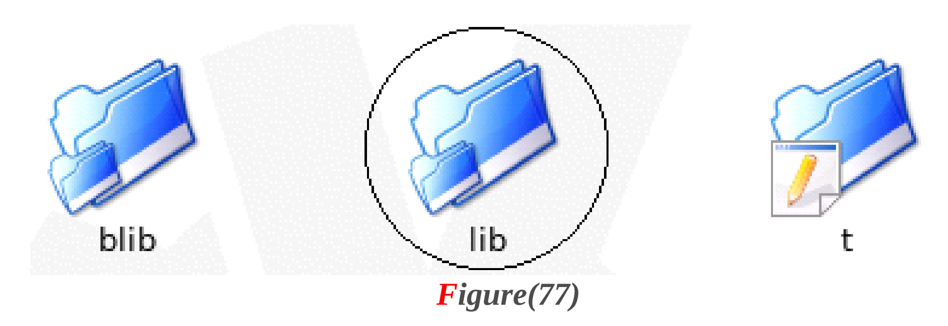
```
use Array::Unique;
tie @a, 'Array::Unique';
@a=("a","b","c","d",1,2,3);
push(@a,"a","c","e",1,2,7,5);
print @a;
```

الان لو تم تنفيذ هذا البرنامج فان ناتجه سوف يعرض كل عنصر مرة واحدة حتى لو تم اخبار المصفوفة من خلال دالة ال push ان يتم اضافة عنصر مكرر الى المصفوفة فانه العنصر هذا لن يتم اضافته وناتج تنفيذ البرنامج هو كما يلي في الصورة الاتية

```
use Array::Unique;
tie @a, 'Array::Unique';
@a=("a","b","c","d",1,2,3);
push(@a,"a","c","e",1,2,7,5);
print @a;
abcd123e75
```

Figure(76)

في هذا البرنامج تم استعمال دالة ال tie لانه هذه الدالة تعمل على ربط متغير وهو @a كما تم في البرنامج الى حزمة او packageلاحظ فقرة مهمة هي انه كل موديل برمجي في كل لغة البيرل يكون مكون من باكيج ولكن عن سبب استعمال هذا النوع من البرمجة في بعض الموديلات وعدم أستخدامه في البعض الاخر ذلك يعود الى كيفية برمجة الموديل الان اذهب الى الموديل array::unique استعرضة سوف تلاحظ انه مكون مما يلي



Figure(77)

الان لو تم فتح هذا المجلد المحاط بالشكل الدائري سوف يكون مكون من مجلد اخر ايضا هو



***F**igure(78)*

و عندما يتم فتح هذا الملف نلاحظ انه يحتوي بداخله على ملف الموديل الفعلي الان سيتم فتح الملف الفعلي للموديل وسيتم ملاحظة مما يتكون هذا الملف

```
package Array::Unique;

use 5.006;
use strict;
use warnings;
use Carp;
```

***F**igure(79)*

كل موديل يتكون من حزمة تشير اليه على كل نحن الان لسنا بصدد برمجة الحزم اذنن سبب استعمال دالة ال tie في البرنامج هو من أجل ان يتم أسناد المتغير الذي كان من نوع مصفوفة الى الحزمة المشار اليها في الشكل اعلاه

P3rL ArrAys

The array trix

اما الان بعد ان تم معرفة كيفية برمجة المصفوفات في لغة البيرل اصبح من المعروف الان لدى المبرمج انه عندما يلاحظ مصفوفة تشبه هذه المصفوفة الموجود في الكود انه يتعامل مع مصفوفة مكونة من 3 عناصر

Code(101)

```
@a=(1,2,3);
```

اما لو كانت لديه مثل هذه

Code(102)

```
@a=(1,2,3,);
```

عند هذه الحالة يتوقع المبرمج انه يتعامل مع مصفوفة مكونة من 3 عناصر معرفة اما العنصر الاخير فهو عنصر null ولكن في الواقع لايوجد عنصر رابع هذه المصفوفة هي تماما كالمصفوفة الموجودة في الكود الاول لايوجد اي فرق برمجي بين هاذين النوعيين من المصفوفات

اذا كان المقطع البرمجي يحتوي على مصفوفة مثل هذه المصفوفة

Code(103)

```
@a=(1,2,3,4,5,6,7);
```

هذه مصفوفة هي مصفوفة عادية مكونة من خمسة عناصر
اما اذا كانت لديك مصفوفة مثل هذه المصفوفة

Code(104)

```
@a=(1..7);
```

اما مصفوفة مثل هذه المصفوفة فلا يجب ان يعتريك الشك على انه هذه مصفوفة مكونة من عنصرين فقط هما الواحد و السبعة بل على العكس هي مصفوفة مكونة من عدد من العناصر وتكون بداية هذه العناصر من الواحد وتنتهي بالرقم 7
و هنا ايضا لايوجد فرق برمجي بين هذين النوعين من البرمجة التنسيقية لعناصر المصفوفة

P3rL HAsh3S

Hashes

النوع الاخير من برمجة المتغيرات في لغة البيرل هي المتغيرات التي تكون من نوع hashes حيث يمثل هذا من المتغيرات بزوج من ال key/value

يكون التمثيل البرمجي العام لبرمجة الهاش كما يلي في الكود الاتي

Code(105)

```
%hash = (
        St0rm          =>"C programmer",
        Striker        =>"Python programmer",
        Mutantion      =>"Ruby programmer",
        Spawn          =>"perl programmer"
        );
```

هذا بالنسبة للتمثيل البرمجي العام للمتغيرات التي تكون من نوع الهاش

اما عن كيفية طباعة هذا النوع المتغيرات فأنه يتم كما يلي من خلال هذا الكود

Code(106)

```
%hash = (
        St0rm          =>"C programmer",
        Striker        =>"Python programmer",
        Mutantion      =>"Ruby programmer",
        Spawn          =>"perl programmer"
        );
print $hash{Spawn};
```

الان لو تم تنفيذ هذا البرنامج فان ناتج تنفيذه يكون كما هو ظاهر في الصورة الاتية

```
%hash = (
        St0rm          =>"C programmer",
        Striker        =>"Python programmer",
        Mutantion      =>"Ruby programmer",
        Spawn          =>"perl programmer"
        );
print $hash{Spawn};
perl programmer
```

Figure(80)

How to print keys

في هذا المثال سوف يتم برمجة الكيفية الخاصة بطباعة المفاتيح التي تكون موجودة ومبرمجة في داخل الهاش وكما يلي من خلال هذا المثال

Code(107)

```
%hash = (
        St0rm        =>"C programmer",
        Striker      =>"Python programmer",
        Mutantion         =>"Ruby programmer",
        Spawn        =>"perl programmer",
        );
print (join("-",keys(%hash)));
print "\n";
```

الان لاحظ لو تم تنفيذ هذا البرنامج فأن ناتج تنفيذه يكون كما يلي في الصورة الاتية

```
St0rm-Mutantion-Spawn-Striker
```

Figure(81)

ومن خلال هذا البرنامج وناتج تنفيذه من الممكن ان نخرج بخلاصة وهذه الخلاصة هي انه في الهاش على الدوام ان القيم التي تكون على اليسار هي التي تكون المفاتيح او ال keys

How to print values

كما تم ذكر انفا في الصفحات القليلة السابقة ان الهاش في لغة البيرل يكون عبارة عن زوج يحمل قيمتين قمية المفتاح وقيمة القيمة وذكرنا كيف يتم طباعة قيمة المفتاح والان حان الوقت لكي يتم ذكر كيف يتم طباعة قيمة القيمة الان سناخذ مثال يشبه المثال السابق مع بعض الاختلاف اليكم الكود

Code(108)

```
%hash = (
        St0rm           =>"C programmer",
        Striker         =>"Python programmer",
        Mutantion               =>"Ruby programmer",
        Spawn           =>"perl programmer",
        );
print (join("-",values(%hash)));
print "\n";
```

الان لاحظ انه لو تم تنفيذ هذا البرنامج فأن ناتج تنفيذه يكون كما يلي في الصورة الاتية

```
C programmer-Ruby programmer-perl programmer-Python programmer
```

Figure(82)

اذن الأن من هذا البرنامج من الممكن ان يتم الخروج بصيغة وهي انه القيم التي تكون على يسار السهم في داخل الهاش هي القيم التابعة للمفاتيح

How to make exist

الان لو تعد بعض صفحة او صفحتين الى الوراء سوف تلاحظ انه تم استعمال هذا الكود في اغلب الحالات

Code(109)

```
%hash = (
        St0rm           =>"C programmer",
        Striker         =>"Python programmer",
        Mutantion               =>"Ruby programmer",
        Spawn           =>"perl programmer",
        );
```

لكن ماذا لو كان هذا الكود مكون من 4 قيم مذا لو كان مكون من 400 قيمة مثلا فأنه من الصعب او المستحيل عليك ان تقرأ جميع هذه العناصر لكي تتأكد فقط من وجود عنصر لذا ففي مثل هذه الحالات فأن لغة البيرل قد جنبتك عناء هذه العملية من خلال استعمال دالة تقوم بهذه العملية وهي دالة الوجودية او دالة ال exists ويكون التمثيل البرمجي العام لهذه الدالة كما يلي ومن خلال هذا الكود الاتي

Code(110)

```
%hash = (
        St0rm           =>"C programmer",
        Striker         =>"Python programmer",
        Mutantion               =>"Ruby programmer",
        Spawn           =>"perl programmer",
        );
if (exists($hash{Spawn})){
print "this key is exists and it`s okk ";
}
```

الان لاحظ انه اذا كانت قيمة هذا المفتاح موجودة في الهاش فأنه سوف يتم طباعة الجملة التي تكون موجودة في السطر الاخير وكما يلي في الصورة الاتية ادناه

```
C programmer-Ruby programmer-perl programmer-Python programmer
```

Figure(83)

ولكن هنالك ملاحظة هامة يجب الاشارة اليها وهي انه في هذه سوف يتم طباعة الجملة الموجودة في الشكل اعلاه مهما كانت القيمة التي تم اسنادها الى المفتاح spawn

How to delete

في هذا الموضوع سوف يتم العمل على مسح مدخل كامل من مداخل العناصر التي تكون موجودة في داخل برمجة الهاش و هذه العملية من خلال استعمال دالة ال delete ويكون التمثيل البرمجي العام لهذه الدالة كما يلي من هذا الكود الاتي

Code(111)

```
%hash = (
        St0rm           =>"C programmer",
        Striker         =>"Python programmer",
        Mutantion               =>"Ruby programmer",
        Spawn           =>"perl programmer",
        );
delete $hash{Spawn};

print %hash,"\n";
```

الان لو تم تنفيذ هذا البرنامج فأن ناتج يكون كما يلي في الصورة الاتية ادناه

```
St0rmC programmerMutantionRuby programmerStrikerPython programmer
```

Figure(84)

الان لاحظ الصورة الموجودة في الاعلى انه عندما تم طباعة الهاش الذي تم ذكره فأنه عندما يتم طباعته سوف كما يقوم بالغاء spawn والقيمة التي تعود لها والتي هي perl programmer كما تلاحظ ناتجة التنفيذ لايحتوي على اي شئ يتعلق ب spawn

ملاحظة هامة جدا هي انه عليك انه تفرق بين عمل دالتين هما دالة ال undef ودالة ال delete
حيث الفرق بين هاتين الدالتين هو انه دالة ال undef فقط تعمل على الغاء التعريف ولا تعمل على الغاء العنصر ولكن دالة delete تعمل على الغاء العنصر بكامله من الهاش

How to each

من أسم الموضوع تلاحظ ان الموضوع يحمل اسم غريب اوغامض نوعا ما يعني ما الذي نعني بعنوان ال how to each من بداية فصل الهاشات ذكرنا انه اذا اردت ان تطبع مفتاح استعمال دالة ال keys واذا اردت ان قيمة تستعمل دالة ال values هذه الدالة تعمل على طباعة القيم التابعة للمفاتيح ولكن ما سوف يحدث لو أردت ان تعمل على طباعة كل من المفتاح و القيمة في نفس الوقت ومن خلال برنامج واحد ومن خلال دالة واحدة اي في هذه الحالة عليك ان تسبعد دوال المفتاح ودوال القيمة لذا في هذه الحالة عليك ان تلجأ الى دالة ال each ومن اسم هذه الدالة تم اشتقاق اسم الموضوع هذه الدالة تعمل على طباعة قيم المفتاح والقيمة من خلال استعمال برنامج واحد فقط ودالة واحدة والتمثيل البرمجي العام لهذه الدالة يتم كما يلي من خلال هذا البرنامج الاتي

Code(112)

```
%hash = (
        St0rm           =>"C programmer",
        Striker         =>"Python programmer",
        Mutantion               =>"Ruby programmer",
        Spawn           =>"perl programmer",
        );
while(($name,$lang)=each(%hash)){
print("$name:$lang","\n");
}
```

الان لاحظ هذا البرنامج وناتج تنفيذه يكون كما يلي في الصورة الاتية المدرجة أدناه

```
St0rm:C programmer
Mutantion:Ruby programmer
Spawn:perl programmer
Striker:Python programmer
```

Figure(85)

عندما تلاحظ ناتج تنفيذ هذا البرنامج فأنه بالفعل دالة ال each تعمل على طباعة قيم المفاتح والقيم المقابلة لها ففي هذه الحالة سوف توفر لك الدالة عناء كتابة برنامجين في برنامج واحد واختصار عمل دالتين في دالة واحدة فقط

How to sort

الدالة التي تستخدم في عمليات الترتيب هي دالة ال sort والتي تم شرحها عملها في الفصول السابقة ولكن هنا
في موضوع برمجة الهاشات الموضوع يختلف من ناحيتين
الناحية الاولى سوف يتم استعمال دالتين
الناحية الثانية الهاشات كما ذكرنا هي عبارة عن ازواج اي مفتاح و قيمة

و الان برمجة القسم الاول تتم كما يلي من خلال هذا الكود الاتي

Code(113)

```
%hash = (
        St0rm           =>"C programmer",
        striker         =>"Python programmer",
        mutantion               =>"Ruby programmer",
        spawn           =>"perl programmer",
        );
print  sort(keys(%hash),"\n");
```

الان لاحظ لو تم تنفيذ هذا البرنامج فأن ناتج التنفيذ سوف يكون مشابه لما موجود في الصورة ادناه

```
St0rmmutantionspawnstriker
```

Figure(86)

في هذا البرنامج تلاحظ انه تم دمج بين دالتين هما دالة الترتيب وهي دالة ال sort والدالة التي تستعمل من أجل
طباعة قيم المفاتيح وهي دالة ال keys ولاحظ حتى في برمجة الهاشات الاحرف ذات الصيغة الكبيرة تكون
لها اسبقية على الاحرف التي تكون لها صيغة صغيرة

How to merge

سيكون المحور البرمجي لهذا الموضوع هو عن كيفية دمج هاش مع هاش وهذه العملية يكون تمثيلها البرمجي كما يلي من خلال هذا الكود

Code(114)

```
%fr34k1 = (
        St0rm  =>"C programmer",
        Spawn =>"perl programmer",
        );
%fr34k2       = (
        Striker        =>"python programmer",
        Mutantion      =>"Ruby programmer",
        );
%fr34ks        = (%fr34k1,%fr34k2);
while (($name,$lang)=each(%fr34ks)){
print ("$name:$lang","\n");
}
```

لاحظ في البرنامج الموجود اعلاه انه لدينا هاشين وكل هاش مكون من عنصرين والخطوة التالية هي يتم اعتبارها

Code(115)

```
%fr34ks        = (%fr34k1,%fr34k2);
```

هذه الخطوة التي تجمع بين الهاشين ويتم توحيدهم في هاش برمجي واحد ومن ثم من أجل طباعة الهاش الذي يحمل الهاشين تم استعمال دالة ال each

والان لو تم تنفيذ هذا البرنامج فأن ناتج تنفيذه سوف يكون كما يلي في الصورة الاتية ادناه

```
St0rm:C programmer
Mutantion:Ruby programmer
Spawn:perl programmer
Striker:python programmer
```

Figure(87)

How to add element

في هذا الموضوع سوف يكون الاتجاه البرمجي في هذا الموضوع هو كيف تقوكم باضافة عنصر مفتاح وقيمة الى الهاش و هذه العملية تتم كما يلي من خلال هذا البرنامج الاتي

code(116)

```
%hash = (
        Storm  =>"C programmer",
        Striker =>"python programmer"
        );
$hash{spawn}="perl programmer";
while (($name,$lang)=each(%hash)){
print ("$name:$lang","\n");
}
```

البرمجة الخاصة بمتغيرات الهاش تختلف عن برمجة المصفوفات حيث بالمصفوفات تتم عن طريقة دالة الدفع او عن طريق دالة unshift
ولكن مع الهاشات كما لاحظنا الوضع يختلف فقط اكتب اسم الهاش والمفتاح وساويه مع القيمة التي تريدها ان تسند اليه ولاحظ ناتج تنفيذ البرنامج يكون كما يلي كما في هذه الصورة الاتية

```
Striker:python programmer
Storm:C programmer
spawn:perl programmer
```

Figure(88)

The Lost Rhyme

***License**

This book is made under the Terms of (GPL) license you may do every thing you want with this Book without an Advance perm you may Edit,add more Chapters or remove antything yo want ,and Redistribute ETC

***Level**

من أجل ان يحصل على قارئ الكتاب على أكثر فائدة علمية من هذا الكتاب فأنه يجب عليه أن يكون على أطلاع بعمل الموديلات و تنصيبها على النظام وعمل الدوال و المتغيرات وعلى المام شامل نوعا ما بالبرمجة في لغة البيرل أما عن هذا الكتاب فأنه بعد ان تم الانتهاء منه نلاحظ انه تناول تقريبا كل شئ عن برمجة المتغيرات في لغة البيرل وكيف يتم التعامل معها واساليب استعمال المتغيرات مع الدوال المبنية في النظام والموديلات الموجودة على الانترنت التي لها علاقة بالمتغيرات

***They**

For every one who stab me in the back believe me am not angry with you i really pity you coz u really pathetic and to the people who thinks that they can devastate my life with there arrogance i guess ur so wrong coz these kinds of cheap attitude ain`t gonna break me.

Friends

(W3bs)

www.programming-fr34ks.net/forum

(programing website and full of programming Fr34ks)

www.securitygurus.net

(security website full of SecUriTy geeks)

(BloGs)

www.linux-fr34k.com

(blog for Linux Security)

www.binary-zone.com

(blog for every thing related to linux)

Special Greet

To My graduation project supervision it`s was a great honor for me to work with you thanks for giving me this honor i really grateful for that and thanks for accepting me as iam god bless you where ever you are and thanks for pushing me forward always and gave me all the support that i want

Gr33tZ

I really would like to thank these people for backing me up and they were very supported friends to me and push me forward.

St0rM_MaN , StrikerX (Ahmed Youssef), SNIX , Blackray ,Mutanti0n , Raidy , sAFA7_eLNET ,Binary

(And Every one in programming-fr34ks.net,securitygurus.net including (Admins & members))

Design

*The Wallpaper of this book has been designed by the inspired design Mr.Sami **SNIX** and this wallpaper is **CopyRight** for The Designer*

NoT3

St0rM_MaN plz wish good luck for him he really need it we all want and all of us wish that you do the best in Exams and get the best marks

Feedback

Feedback,Ideas,Bugs,any thing else send it to

E-Mail:- *Mahmoud.Najafy@Gmail.com*


WrOtE BY:-
M_SpAwN
(SaXaPhOnE PlAyEr)